KAUNO TECHNOLOGIJOS UNIVERSITETAS

ELEKTROS IR ELEKTRONIKOS FAKULTETAS

Julius Žemgulys

KREPŠINIO TEISĖJO ŽENKLŲ ATPAŽINIMO METODO SUKŪRIMAS IR TYRIMAS

Baigiamasis magistro projektas

Vadovas
Doc. dr. Vidas Raudonis

KAUNAS, 2018

KAUNO TECHNOLOGIJOS UNIVERSITETAS

ELEKTROS IR ELEKTRONIKOS FAKULTETAS
AUTOMATIKOS KATEDRA

KREPŠINIO TEISĖJO ŽENKLŲ ATPAŽINIMO METODO SUKŪRIMAS IR TYRIMAS

Baigiamasis magistro projektas
Valdymo technologijos (kodas 621H66001)

Vadovas
Doc. dr. Vidas Raudonis

Recenzentas
Doc. dr. Virginijus Baranauskas

Projektą atliko
Julius Žemgulys

KAUNAS, 2018

KAUNO TECHNOLOGIJOS UNIVERSITETAS

Elektros ir elektronikos

(Fakultetas)

Julius Žemgulys

(Studento vardas, pavardė)

Valdymo technologijos, 621H66001

(Studijų programos pavadinimas, kodas)

Baigiamojo projekto „Krepšinio teisėjo ženklų atpažinimo metodo sukūrimas ir tyrimas“

AKADEMINIO SĄŽININGUMO DEKLARACIJA

20 18 m. gegužės 25 d.

Kaunas

Patvirtinu, kad mano **Juliaus Žemgulio** baigiamasis projektas tema „Krepšinio teisėjo ženklų atpažinimo metodo sukūrimas ir tyrimas“ yra parašytas visiškai savarankiškai, o visi pateikti duomenys ar tyrimų rezultatai yra teisingi ir gauti sąžiningai. Šiame darbe nei viena dalis nėra plagijuota nuo jokių spausdintinių ar internetinių šaltinių, visos kitų šaltinių tiesioginės ir netiesioginės citatos nurodytos literatūros nuorodose. Įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs.

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

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

Julius, Žemgulys. Krepšinio teisėjo ženklų atpažinimo metodo sukūrimas ir tyrimas. *Valdymo sistemų magistro* baigiamasis projektas / vadovas doc. dr. Vidas Raudonis; Kauno technologijos universitetas, Elektros ir elektronikos fakultetas, Automatikos katedra.

Mokslo kryptis ir sritis: Elektros ir elektronikos inžinerija, Technologiniai mokslai

Reikšminiai žodžiai: *vaizdų atpažinimas, pirminis apdorojimas, klasifikavimas, bruožų išskyrimas*.

Kaunas, 2018. 66 p.

SANTRAUKA

Šio darbo tikslas – sukurti tesėjo gestų atpažinimo sistemą, naudojant skirtingas bruožų išskyrimo ir klasifikatorių kombinacijas, bei ištirti sukurtą sistemą.

Pirmame baigiamojo darbo skyriuje, atliekama literatūros apžvalga, siekiant nustatyti išankstinio apdorojimo, bruožų išskyrimo ir klasifikavimo metodus, tinkančius gestų atpažinimo užduočiai. Pateiktos matematinės metodų išraiškos, kurios padeda geriau suvokti skirtingų metodų procesus.

Antroje dalyje apžvelgiamas sukurtas krepšinio teisėjo ženklų atpažinimo sistemos algoritmas. Algoritmas buvo realizuotas naudojant „MATLAB“ programinį paketą. Buvo sudaryta duomenų bazė iš 3000 nuotraukų, kur teisėjas rodo gestus, po 500 nuotraukų, šešioms gestų klasėms. Sistema realizuota 7 metodais: naudojant HOG, LBP ir PCA bruožų išskyrimo metodus su SVM ir RF klasifikatoriais, taip pat apmokytas CNN klasifikatorius.

Trečiame skyriuje atliekami suskurtos sistemos tyrimai. Svarbiausi realizuotos sistemos parametrai: klaidingai aptiktų komandų skaičius, tikslumas, klasifikatoriaus apmokymo laikas ir klasifikavimo trukmė. Sistema buvo testuojama su 60 testavimo nuotraukų, kuriose teisėjas rodo skirtingus ženklus, taip pat buvo naudojamos trijų skirtingų dydžių nuotraukos klasifikatorių apmokymui. Didžiausias pasiektas atpažinimo tikslumas 85%, naudojant atsitiktinių miškų klasifikatorių ir orientuotų gradientų histogramą.

Julius Žemgulys. RESEARCH AND DEVELOPMENT OF BASKETBALL REFEREE SIGN RECOGNITION SYSTEM*: Master`s* thesis in *Control System* / supervisor doc. dr. Vidas Raudonis. Kaunas University of Technology, Faculty of Electrical and Electronics Engineering, Department of Automation.

Research area and field: Electrical and Electronics Engineering, Technological Sciences

Key words: *image processing, pre-processing, feature extraction, classification.*

Kaunas, 2018. 66 p.

SUMMARY

The aim of this work is to create a system for recognizing gestures of the basketball referee, using different combinations of feature extraction and classification algorithms, and to analyze the created system.

In the first section of the final work, a review of the literature was carried out in order to determine the methods for pre-processing, distinguishing features and classification suitable for the gesture recognition task. Mathematical expressions of methods are presented, which will help to understand the processes of different methods.

In the second part the developed algorithm of the basketball referee sign recognition system is reviewed. The algorithm was implemented using the MATLAB software package. A database of 3000 photos was created, where the judge displays gestures, hence 500 photographs for one class, six gesture classes. The system was implemented in 7 methods: using the HOG, LBP, and PCA features extraction methods with SVM and RF classifications, and also trained using the CNN classifier.

In the third chapter, investigations of the created system are carried out. The most important parameters of the implemented system are the number of mistakenly detected commands, accuracy, classification training time and classification time. The system was tested with 60 test photos, in which the judge displayed different signs, and also three different sizes photos were used for treining the classifier. The best-known recognition accuracy is 85%, using a histogram of oriented gradients and random forests classification algorithm.

Turinys

SANTRUMPŲ ŽODYNAS

- SVM (angl. *Support vector machine*) – atraminio vektoriaus mašinas;
- RF (angl. *Random forests*) – atsitiktinių miškų algoritmas;
- CNN (angl. *Convoliutonal neural network*) – konvoliucinis neuroninis tinklas;
- PCA (angl. *Principal Component Analysis*) – principinė komponentų analizė;
- HOG (angl. *Histogram of Oriented gradients*) – orientuotų gradientų histograma;
- LBP (angl. *Local Binary Pattern*) – lokali binarinė seka;
- HSV (angl. *Hue, Saturation, Value*) – atspalvis, įsisotinimas, vertė.

ĮVADAS

Gestų atpažinimas, populiarus uždavinys skaitmeninio vaizdų apdorojimo srityje. Dažniausiai šis metodas naudojamas žmogaus ir kompiuterio sąveikai realizuoti. Krepšinio teisėjo gestai skirti krepšinio sekretoriatui ir žaidėjams. Krepšinio sekretoriatas, nevisada teisingai užregistruoja teisėjo rodytas komandas, dėl to tenka stabdyti varžybas arba šios klaidos pastebimos jau po varžybų. Dėl žmogiškojo faktoriaus klaidos eliminavimo, buvo nuspręsta sukurti krepšinio teisėjo gestų atpažinimo sistemą, kuri galėtų būti naudojama, kaip papildomas įrankis sklandžiai varžybų eigai užtikrinti.

Skaitmeninis vaizdų apdorojimas – kompiuterio vaizdų analizės, sustiprinimo, kompresijos bei atkūrimo būdų rinkinys. Vaizdų apdorojimo procesas gali būti apibūdinamas trimis žingsniais:

1. Vaizdo įkėlimas naudojant optinį skaitytuvą ar skaitmeninę nuotrauką;
2. Vaizdo analizavimas ir valdymas – duomenų kompresija, vaizdų stiprinimas (norimo atpažinti objekto ryškinimas...) ir bruožų išskyrimas;
3. Sistemos išėjimas (rezultatai), atlikus vaizdo analizę, tai gali būti pakeistas vaizdas ar sugeneruotas, analize pagrįstas, raportas.

Pagrindinės sritys, kuriose yra taikomas skaitmeninių vaizdų apdorojimas:

- Išmaniosios transporto sistemos – automatinis numerių, ženklų atpažinimas, kliūčių fiksavimas, eismo juostos atpažinimas, aklos zonos stebėsena ir kt.
- Nuotolinis registravimas – nufotografuojamas žemės paviršius iš satelitų arba naudojant specialius spektrinius jutiklius su lėktuvais. Vaizdai apdorojami ir gauta informacija gali būti naudojama potvynių kontrolei, miestų planavimui, resursų mobilizacijai, žemės ūkio stebėsenai ir kt.
- Judančių objektų sekimas – galima matuoti judėjimo parametrus ir gauti vaizdinį judančio objekto įrašą.
- Gynybinė stebėsena – skenuojant aplinką iš lėktuvo, sunku pastebėti tam tikrus objektus didelėse teritoriose. Apdorojant vaizdus, galima išskirti tam tikrus objektus ir tai palengvina saugumo užtikrinimą.
- Biomedicina – naudojant skirtingus vaizdavimo įrankius – rentgeno spindulių, ultragarso, tomografijos daroma medicininė diagnostika.
- Automatinės vizualinės apžiūros sistemos – taikoma norint užtikrinti geresnę produktų kokybę ir gamybos našumą įvairiose pramonės srityse.

Tyrimo tikslas

Sukurti krepšinio teisėjo gestų atpažinimo sistemą naudojant skirtingas klasifikatorių/bruožų kombinacijas, jas tarpusavyje palyginti ir ištirti šios sistemos tikslumą.

Tyrimo uždaviniai

1. Išanalizuoti literatūros šaltinius. Apžvelgti vaizdų klasifikavimo, išankstinio aborojimo ir bruožų išskyrimo metodus.
2. Sukurti krepšinio teisėjo komandas atpažįstančią sistemą, naudojant skirtingas klasifikatorių, bruožų išskyrimo ir išankstinio apdorojimo kombinacijas.
3. Palyginti skirtingų algoritmų veikimą (klaidingų komandų fiksavimo skaičius, neužfiksuotos komandos ir kt.).
4. Pasiūlyti apibendrintus sprendimus krepšinio komandų atpažinimo sistemos tobulinimui.

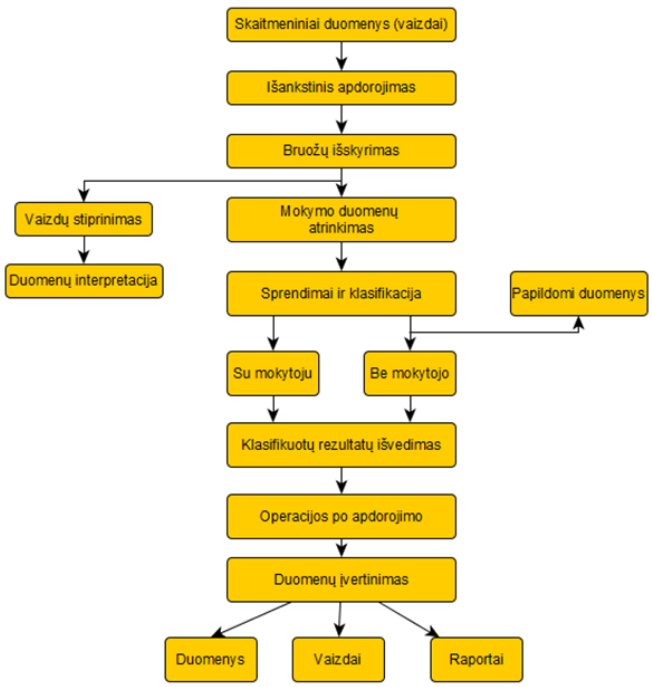
1. LITERATŪROS APŽVALGA

1.1. Vaizdų atpažinimo sistema

Vaizdas – dviejų dimensijų funkcija f(x, y), kur x ir y yra erdvinės koordinatės, o f – amplitudė, vadinama vaizdo intensyvumu bet kurioje vaizdo vietoje x ir y koordinačių atžvilgiu. Kai yra baigtinis kiekis x, y ir f intensyvumo lygių verčių, vaizdas vadinamas skaitmeniniu. Dažnai tenka apdoroti vaizdus, kurie turi ir daugiau nei dvi dimensijas pavyzdžiui: f(x, y, ʎ) – spektrinis vaizdas su tam tikru kiekiu skirtingų bangos ilgių, f(x, y, t) – laike kintantis nespalvotas dvimatis vaizdas, f(x, y, z, ʎ, t) – laike kintantis spalvotas trimatis vaizdas (realybė) [2].

Skaitmeninis vaizdas yra sudarytas iš tam tikro elementų skaičiaus, kiekvienas iš jų turi skirtingas koordinates ir nevienodas vertes. Šie elementai vadinami pikseliais. Šių elementų išskyrimas ir apdorojimas, naudojant skaitmeninį kompiuterį, yra skaitmeninio vaizdų apdorojimo pagrindas. Kompiuterio matymo sistemos techninė įranga yra sudaryta iš kameros, kameros sąsajos ir kompiuterio [1].

Visos matymo ar vaizdų analizavimo sistemos turi bazinę programinės įrangos infrastruktūrą. Kuriant naują vaizdų atpažinimo sistemą, apdorojami vaizdai, jų pikseliai, kurių gali būti daugiau nei 6 milijonai ir jiems suteikiamos tam tikros vertės. Apmokius sistemą ir atlikus duomenų klasifikaciją, gauta informacija apdorojama ir išvedamas sistemos rezultatas. Vaizdų atpažinimo sistemos algoritmas pateikiamas 1.1 pav. [3].



1.1 pav.Vaizdų atpažinimo sistemos algoritmas

Pirminis skaitmeninių duomenų apdorojimas – šis sistemos realizavimo etapas priklauso nuo pasirinkto bruožų išskyrimo metodo ir vaizdų, skirtų sistemai apmokyti, kokybės. Populiariausi išankstinio apdorojimo metodai – susiliejusių vaizdų korekcija, kontrasto padidinimas, vaizdo dydžio keitimas, išplėtimas ir karpymas.

Bruožų išskyrimas – tai vienas svarbiausių etapų vaizdų atpažinimo sistemose. Svarbu išskirti ir atrinkti tinkamus bruožus, kurie bus naudojami klasifikacijos ir duomenų įvertinimo etapuose. Pagrindinis bruožų išskyrimo tikslas – išskirti aktualiausią informaciją iš visų duomenų tam tikrais paprastais skaičiais. Plačiai naudojami bruožų išskyrimo metodai: šablonų atitikimo tikrinimas, deformuojami šablonai, vaizdų transformacijos, vaizdo apibūdinimas kreive, projekcijų histogramos, kontūrų profilių išskyrimas, skirstymas į zonas, geometrinių momentų variantai, *Zernik*e momentai, *Spline* kreivės aproksimacija, *Furje* aprašymai, *Gabor* ir gradientų charakterizavimas. Šio projekto realizacijai svarbu išskirti teisėją iš fono ir tada atrinkti visas teisėjo komandas.

Vaizdų stiprinimas ir duomenų interpretacija yra taikomi ne visose vaizdų atpažinimo sistemose. Duomenų interpretacija reikalinga, kai nėra tikslios informacijos apie apdorojamus duomenis. Vaizdų stiprinimas gali būti atliekamas erdvės ir dažnių srityse, siekiant pagerinti vaizdo kokybę, jei to nepavyko padaryti išankstinio apdorojimo fazėje.

Duomenų klasifikacija – ankstesniuose vaizdų atpažinimo sistemos etapuose išskirtų duomenų suskirstymas į kategorijas pagal tam tikrus jų panašumus. Klasifikuojant vaizdus taikomi du pagrindiniai metodai – klasifikacija su mokytoju ir be. Klasifikuojant su mokytoju reikalingas klasifikatoriaus apmokymas, suskirstant ir priskiriant duomenis skirtingoms sritims. Naudojant klasifikatorių be mokytojo, duomenys surenkami atsitiktine tvarka ir suskirstomi pagal tam tikrus panašumus. Klasifikacijos su mokytoju pranašumai – operatoriai dažnai randa ir pašalina klaidas, reikalingas duomenims įvertinti, todėl naudojant šį metodą galima pasiekti tikslesnių rezultatų. Trūkumai – netinkamas dideliam duomenų kiekiui apdoroti, kadangi ilgai užtrunka, kol duomenys priskiriami konkrečioms sritims. Klasifikuojant be mokytojo, nereikalingas duomenų skirstymas į tam tikras sritis, todėl šis metodas yra greitas ir tinkamas apdorojant didelius duomenų kiekius. Taikant klasifikacijos be mokytojo metodą, reikia labai gerų žinių apie tiriamąją sritį, kadangi nereikia apmokyti klasifikatoriaus. Šiuo metodu apdorojant didelius duomenų kiekius, skaičiavimo laikas yra ilgas. Pagrindiniai klasifikavimo metodai yra dirbtinis neuroninis tinklas (ANN), tiesioginė gaubtine kreive pagrįsta atraminio vektoriaus mašina (DAG-SVM), atraminio vektoriaus mašina (SVM), neraiškios logikos sprendimų medis (FDT), „Random Forests“ ir t. t. [5].

1.2. Pirminis vaizdų apdorojimas

Žmogaus vaizdų suvokimas yra apbrėžiamas dviem pagirindiniais parametrais: spalva ir apšvietimu. Skaitmeniniame vaizde šie parametrai apibrėžiami tam tikro spalvų spektro pikselių intensyvumu. Visi išankstinio apdorojimo metodai yra paremti spalvų paletės ir pikselių intensyvumo korekcija arba keitimu.

1.2.1.Spalvos

Norint išskirti tikslius norimo atpažinti objekto bruožus, dažnai reikalingas spalvų paletės konvertavimas. Keturi dažniausiai naudojami spalvų modeliai: RGB, CMY, HSI ir YC_bC_r. Modeliai gali būti konvertuoti iš vieno į kitą, tam reikalingi matematiniai ar trigonometriniai perskaičiavimai. Spalvos yra koduojamos bitais, pavyžđžiui 8 bitų RGB modelio B dedamosios vertės yra nuo 0 iki 255 [12].

RGB spalvų paletę sudaro trys pirminės spalvos: raudona, žalia ir mėlyna. Skirtingomis šių spalvų kombinacijomis galima gauti norimus atspalvius. Nepaisant to, kad tai dažniausiai naudojama paletė vaizdų technikoje, ji nėra tinkama vaizdų apdorojimo algoritmui, nes jos dedamosios tarpusavyje koreliuoja. Vaizdų apdorojimui dažniausiai naudojami juodos ir baltos spalvos vaizdai su pilkais pustoniais. Perskaičiavimas iš RGB į nespalvotą paletę:

$$JB = 0{,}333R + 0{,}333G + 0{,}333B. \tag{1.1}$$

CMY spalvų modelį sudaro žydra, raudonai rausva ir geltona spalvos. Šis spalvų modelis yra laikomas absorbuojančiu šviesą, todėl dažniausiai naudojamas spausdintuvuose. Konversija iš RGB į CMY išreiškiama nesudėtingomis formulėmis:

$$C = 1.0 - R, \qquad M = 1.0 - G, \qquad Y = 1.0 - B. \tag{1.2}$$

HSI spalvų modelis yra sukurtas pagal tris parametrus: saturacija, apšvietimas ir atspalvis. Naudojant šį modelį spalvą galima nustatyti pasirinkus atspalvį, tada spalvos sodrumą, bei apšvietimą. Norint atpažinti objektą pagal jo spalvą, atlikti histogramos ar vaizdo intensyvumo pakeitimus , naudojama HSI spalvų paletė. Atspalvis (H) yra išreikštas laipsniai ir kinta nuo 0^o iki 360^o , saturacija (S) ir apšvietimas kinta intervale nuo 0 iki 1. HSI modelis aprašomas balta ir juoda spalvomis su pilkai atspalviais. RGB konvertavimas į HSI:

$$I = \frac{1}{3}(R, G, B); \tag{1.3}$$

$$S = 1 - \frac{3}{R+G+B}[\min(R,G,B)];$$

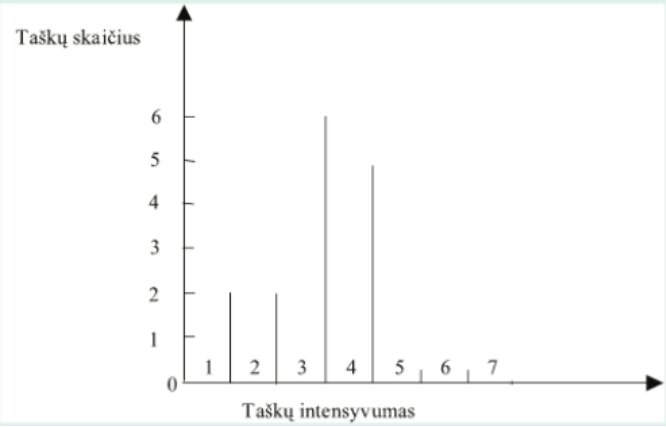
$$H = \cos^{-1}[\frac{\frac{1}{2}[(R-G)+(R-B]}{\sqrt{(R-g)^2+(R-B)(G-B)}}].$$

Jeigu vaizdų apdorojimo algoritme reikia eliminuoti apšvietimo įtaką, reikalingas spalvų konvertavimas į YC_bC_r paletę, nes ją sudaro monochromatinės spalvos. Konvertavimas iš RGB į YC_bC_r [12]:

$$\begin{aligned} Y &= 0{,}299R + 0{,}587G + 0{,}144B; \\ Cb &= -0{,}16874R + 0{,}33126G + 0{,}5B; \\ Cr &= -0{,}5R + 0{,}41869G + 0{,}08131B. \end{aligned} \tag{1.4}$$

1.2.2. Operacijos su histogramomis

Histograma – grafikas, kuriame nurodomas taškų kiekis ir jų intensyvumas paveiksle. Naudojantis histograma galima nustatyti vaizdo kontrastingumą ir taškų intensyvumą. Histogramos grafikas yra sudarytas iš dvimatės XY plokštumos, y ašyje nurodytas taškų inetnsyvumas, x taškų skaičius 1.2 pav. [12].



1.2 pav. Vaizdo histograma

Paveikslo pikselio intensyvumai histogramoje sugrupuojami didėjančia išsidėstymo tvarka. Taško intensyvumo stulpelio dydis priklauso nuo tam tikro intensyvumo taškų kiekio paveikslėlyje. Iš histogramos papsiskirstymo galima spręsti apie paveikslėlio kokybę. Tolygus histogramos pasiskirstymas reiškia, kad paveikslas subalansuotas, jeigu matomas „pikas“ paveikslas yra nekontrastingas, o jei matoma gaubtinė, paveiksliukas yra labai kontrastingas [12].

Histogramų pagrindu dažnai atliekamos vaizdų korekcijos, siekiant pagerinti bruožų išskyrimo ir klasifikavimo etapus. Pagrindinės transformacijos: gama korekcija, kontrasto keitimas, histogramos tankio lyginimas, binarizavimas, inversija, intensyvumo lygio išpjovimas.

Atliekant gamos korekciją netiesiškai pakeičiamas paveikslo šviesumo lygis ir išryškinamos detalės. Gama korekcija reikalinga, norint kompensuoti netiesinius iškraipymus, susidariusius dėl vaizdų skaitmenizavimo. Gama korekcija apskaičiuojama:

$$O = I^{\frac{1}{\gamma}}, \tag{1.5}$$

kur I – normalizuota taško reikšmė, O – perskaičiuota normalizuota taško reikšmė, γ – gama koeficientas. Pagal γ vertę galima nustatyti ar paveikslas yra tamsinamas ar šviesinamas, jei γ > 1 šviesinamas, jei 0 < γ < 1 – tamsinamas, o kai γ = 1 – nesikeičia [12].

Kontrasto keitimas – tiesinė paveikslėlio histogramos transformacija, kurios metu keičiamas vaizdo kontrastas, t.y. atskiriami šviesūs tonai nuo tamsių. Kontrastą galima keisti tiesiškai ir netiesiškai. Kontrastą keičiant netiesiškai, skiritngiems intervalams naudojami skirtingi kontrasto skaičiavimo koeficientai L ir H. Tolydaus kontrasto keitimo formulė:

$$O = \frac{I - L}{H - L} n, \tag{1.6}$$

kur I – orginali taško reikšmė, O – perskaičiuota taško reikšmė, L – mažiausia taško reikšme visame diapazone, H – didžiausia taško reikšmė, n – maksimali galima taško vertė.

Histogramos tankio lyginimas naudojamas paveikslo kontrasto padidinimui, po išlyginimo vyrauja pilki pustoniai. Po šios operacijos gaunama tolygiai pasiskirčiusi histograma. Histogramos tankio išlyginimas aprašomas formule (naudojami parametrai sutampa su nurodytais ankstesnėse formulėse):

$$O = \begin{cases} 0, jei\ I < L \\ \frac{I - L}{H - L} n, jei\ L < I < H. \\ n, jei\ L > H \end{cases} \tag{1.7}$$

Binarizavimas – labai supaprastina bruožų išskyrimo etapą, naudojant šį metodą vaizdas transformuojamas į juodos ir baltos spalvos (binarinį) vaizdą. Binarizavimas atliekamas iš vaizdo su pilkais pustoniais, parenkant tam tikrą slenksčio ribą, jei taškas yra žemiau užduoto slenksčio jis prilyginamas 0 (juoda), jei taškas aukščiau už slenkstį taškas prilyginamas 1 (balta).

Inversija – histogramos transformacija kai sukeičiami paveikslo tamsūs ir šviesūs pustoniai, inversija išreiškiama formule:

$$O = n - I. \tag{1.8}$$

Intensyvumo lygio išpjovimas – histogramos transformacijos funkcija, kurios metu pašalinami vaizdo taškai, esantys žemiau parinktos slenksčio ribos. Jeigu norima palikti vaizdo foną, taškų vertės esančios žemiau už slenkstį yra prilyginamos pradinei taško vertei [12].

1.3. Bruožų išskyrimo metodų apžvalga

Bruožų išskyrimo metodai pasirenkami priklausomai nuo realizuojamos sistemos tipo. Pagrindiniai objekto bruožai skirstomi į tris grupes: žemo lygio bruožai, aukšto lygio bruožai ir nenustatomi objekto parametrai. Žemo lygio bruožai gali būti išskiriami neturint jokios informacijos apie norimą atpažinti objektą. Tokie bruožai gali būti objekto kraštai, kampai, taip pat galima aptikti objekto judėjimo seką, naudojant optinės tėkmės metodą, fiksuojant tam tikrų taškų koordinačių pasikeitimą. Pagrindinis objekto aukšto lygio bruožas yra jo forma. Pastaroji turi būti pastovi ir aiškiai išskirta iš fono, gali keistis jos pozicija, pasisukimo kampas, dydis (atstumas nuo kameros). Kai tiksliai nežinoma objekto forma ir nėra kitų parametrų iš kurių objektas galėtų būti atpažintas, reikalinga atpažinimo technika, kuri prisitaikytų prie kintančių objekto parametrų [6].

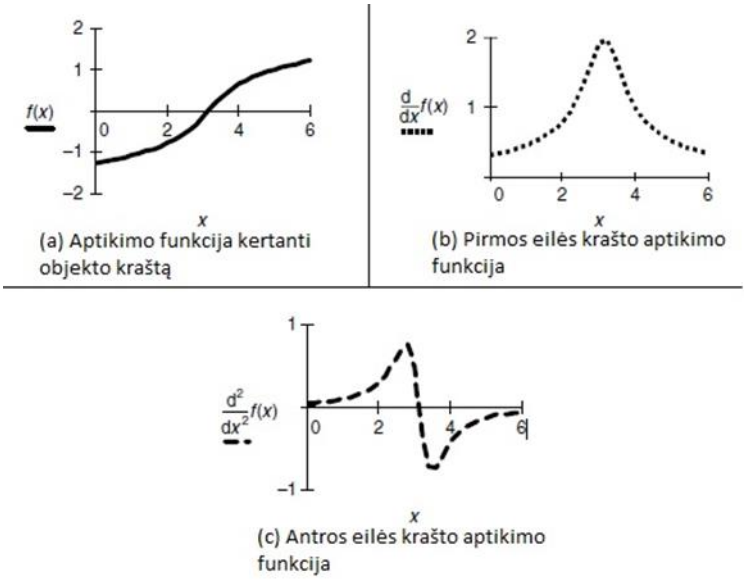
1.3.1. Bendri bruožų išskyrimo metodai

Daugelyje sistemų taikomas objektų bruožų išskyrimas, pagal kraštus. Taikant šį metodą fono apšvietimo pasikeitimas didelės įtakos neturi. Kraštų išskyrimas pabrėžia vaizdo kontrastą. Suradus žingsnio vertę, per kurią objekto intensyvumo lygis skiriasi nuo aplinkos, galima nustatyti objekto kraštą. Norint aptikti objekto krašto poziciją naudojama pirmos eilės diferenciacija, intensyvumo pokytis aptinkamas diferencijuojant artimus taškus. Yra taikomi horizontalių (Ex) ir vertikalių (Ey) kraštų detektoriai, skirtingose vaizdo koordinatėse, kai vaizdas yra žymimas P, detektoriai aprašomi formule (1.9):

$$Ex_{x,y} = \left|P_{x,y} - P_{x+1,y} + P_{x,y} - P_{x,y+1}\right|, \forall x, y \epsilon 1, N-1. \quad (1.9)$$

Pirmos eilės kraštų aptikimo metodas sukurtas remiantis prielaida, kad vaizdo intensyvumas keičiasi objekto ribose. Procesas vaizduojamas 1.3 pav., kur 1.3 pav. a) yra funkcija kertanti norimo atpažinti objekto skerspjūvį. Pirmos eilės aptikimo diferencialinės lygties $f'(x) = df/dx$ rezultatas vaizduojamas 1.3 pav. b, kuriame aukščiausias taškas yra funkcijos 1.3 pav a) didžiausias pokytis. Objekto aptikimui gali būti taikomos ir aukštesnės eilės

išvestinės $f''(x) = d^2f/dx^2$ (1.3 pav. c), kur šio signalo pokytis yra didžiausias, ten kertamas objekto skerspjūvis, o kur pokytis mažiausias ten originalaus signalo pokytis pastovus. Taigi nulio kirtimo (objekto krašto) fiksavimui, priklausomai nuo norimo atpažinti objekto, gali būti taikomos pirmos arba antros eilės diferencialinės lygtys.



1.3 pav.Pirmos ir antros eilės krašto aptikimo funkcijos

Norint aptikti judėjimą vaizde, reikia daugiau nei vienos nuotraukos. Turint du vaizdus, kurie buvo užfiksuoti skirtingais laiko momentais, jų judėjimą galima aptikti juos diferencijuojant. Pokyčiai arba judėjimas gali būti užfiksuoti naudojant šių paveikslėlių pikselių intensyvumo atimtį, jei objektas nejuda, atimties rezultatas bus 0. Vaizdų sekos nurodymui turi būti įvertintas laikas. Skirtumas D tarp dviejų vaizdų skirtingais laiko momentais, kai paveikslėliai yra aprašomi funkcijomis $P(0)_{x,y}$ ir $P(1)_{x,y}$, yra aprašomas lygtimi (1.10):

$$D(t) = P(t) - P(t-1). \tag{1.10}$$

Aukšto lygio bruožų išskyrimas kompiuterio vaizde tai formų suradimas. Pavyzdžiui norint automatiškai atpažinti veidus, galima išskirti veido bruožų sudedamąsias dalis. Pagrindinės veido dalys turi skirtingas formas: burna – tarsi dvi linijos, akių vyzdžiai – apskritimai ir t. t. Daugelyje sistemų galima įvertinti ne tik kokios formos turėtų būti vaizde, bet ir kokioje pozicijoje jos yra viena kitos atžvilgiu. Bruožų išskyrimo etape, svarbu, kad vaizduose būtų nekintamų parametrų, nepriklausomai nuo formų dydžio ar apšvietimo nuotraukoje. Svarbu įvertinti objekto rotacijos ir dydžio pokyčius. Reikalingas ryškus kontrastas tarp objekto ir fono, todėl šiam bruožų išskyrimo metodui yra labai svarbus išankstinio apdorojimo etapas.

Paprasčiausia objekto figūros išskyrimo technika yra slenksčio parinkimas, apdorojus norimą atpažinti objektą taip, kad jis būtų ryškesnis už aplinką. Kai objektą apibrėžia jo ryškumas, svarbu parinkti tinkamą slenkstį ten, kur funkcija kirs slenksčio ribą ir bus galima teigti, kad ten yra norima atpažinti figūra. Šis metodas yra labai jautrus fono apšvietimo pasikeitimams, todėl jo esmė – keisti slenksčio ribą, priklausomai nuo vaizdo apšvietimo pasikeitimo. Taikant šį metodą nereikia atlikti daug skaičiavimų, todėl jis yra palyginti nesudėtingas ir tai yra pagrindinis jo pranašumas. Jei vaizdo apšvietimo lygis kinta tiesiškai, užtenka naudoti rezultatų išlyginimo histogramą ir galima gauti vaizdą, kurio apšvietimas nekinta. Histogramos trūkumai – nekompensuojamas triukšmas, šešėliai ir netiesiškas apšvietimo kitimas, todėl dažnai naudojamas formos iškirpimas iš fono, prieš nustatant slenkstį. Jei bent vienas vaizdas (iškirpta figūra ir bendras vaizdas) yra triukšmingas, objekto atpažinimas tampa beveik neįmanomu [6].

Paveikslėlyje (*žr.* 1.4 pav.) siekiama išskirti judantį objektą iš fono. Iš vaizdo (a) yra iškerpamas judantis žmogus (b), tačiau už jo galvos yra šiek tiek fono, taip yra dėl judančio objekto poveikio apšvietimui. Galutinis vaizdas matomas nuotraukoje (c), kur parinkus slenkstį objektas paryškinamas ir dingsta kai kurie jo parametrai (linijos ant marškinėlių).

1.4 pav.Objekto formos atpažinimas atimant foną

Kita aukšto lygio bruožų išskyrimo technika yra vaizdo lyginimas su šablonu. Šablone yra tik norimas atpažinti objektas ar jo forma, o visuose vaizdo taškuose tikrinama kiek taškų sutapo lyginant su šablonu. Radus visus pikselių sutapimus vaizde, nustatoma objekto pozicija. Šis procesas gali būti pritaikytas skirtingu kampu pasisukusiems objektams rasti, tada reikia daugiau apmokymo medžiagos (šablonų) ir laiko. Šiuo metodu yra vertinami keli parametrai – objekto pozicija ir pasisukimo kampas. Šablonas gali būti aprašytas diskrečia funkcija Tx,y. Šiai funkcijai reikalingos taškų koordinatės (x, y) ϵ W. Pavyzdžiui 2x2 šablono taškai bus W = { (1,1), (1,0), (0,1), (0,0)}. Vėliau tikrinant vaizdą pasirinktu metodu ir sudarius atitinkamas lygtis, ieškomas šablono pikselių atitikimas vaizde [6].

Hough transformacija – tai technika, išskirianti linijas, apskritimus ir elipses iš vaizdo. Šis metodas buvo pristatytas 1962 m., kai vietoj to, kad vaizde būtų ieškomą formų, buvo ieškoma apskritimų sekų. Vėliau buvo ieškoma tam tikrų linijų sekų vaizduose ir buvo nustatyta, kad šis metodas yra panašus į šablonų atitikimo, tik daug spartesnis. Sparta yra pasiekiama performuluojant šablonų atitikimo procesą. Taikant HT metodą figūrų išsidėstymas iš vaizdo perkeliamas į kaupiamąjį duomenų masyvą. Figūrų išsidėstymas, atitinkantis objekto formą, sukuriamas naudojant efektyvius skaičiavimus. Metodui realizuoti reikia mažiau kompiuterio skaičiavimo resursų, nei vaizdo lyginimui su šablonu. Todėl HT metodas optimizuojamas ir jo funkcionalumas plečiamas stengiantis išlaikyti jo charakteristikas, išsaugojant objektų išskyrimo greitį [6].

Kai nežinoma tiksli objekto forma arba ji yra kintanti, reikalingas bruožų išskyrimo metodas, kuris gali evoliucionuoti ir išskirti skirtingus bruožus priklausomai nuo duomenų arba gautus rezultatus pritaikyti esamiems duomenims. Lanksčių formų išskyrimui iš vaizdo naudojamos keturios pagrindinės technikos: lanksčių šablonų taikymas, aktyvūs kontūrai ir vingiai, formų vertimas karkasais, aktyvių formų modeliai. Lanksčių šablonų taikymas, kai ieškoma atitikimų tarp turimų šablonų ir kintančių objekto formų vaizde. Aktyvūs kontūrai ir vingiai – tai objektų ieškojimas evoliucionuojant kontūrus, gali būti diskrečių ir nepertraukiamų formuluočių. Formų vertinimas karkasais – įvertinami atstumo, karkaso ir simetrijos parametrai, simetrijos aptikimas kaupiant parodymus. Aktyvių formų modeliai – formos kitimas apibūdinamas statistiniais duomenimis [6].

1.3.2. Orientuotų gradientų histograma (HOG)

Krepšinio teisėjo gestų atpažinmo sistema, iš esmės yra objektų atpažinimo sistema, ieškomas objektas vaizde bus teisėjo gestas. Klasikinis objektų atpažinimo uždavinys – nurodyti ar vaizde yra žmogus. Sudarant treniravimo duomenų bazę objektų detektavimui, svarbu išlaikyti panašų vaizdo santykį, pavyzdžiui stovinčio žmogaus vaizdo santykis 1:2, tai reiškia kad treniravimo paveikslėlio plotis turi būti dvigubai mažesnis už aukštį (128x64 pikselių arba 64x32 pikselių). Toks langas slenka per vaizdą, kuriame turėtų būti norimas atpažinti objektas, tam tikru žingsiu vertikaliai, bei horizontaliai ir kiekviename lange sugeneruojamas bruožų vektorius, kuris vėliau bus naudojamas klasifikacijai. Kadangi langas yra fiksuoto dydžio, o norimas atpažinti objektas gali būti didesnis arba mažesnis, todėl reikia slinkti langą pro skirtingų matmenų vaizdų piramidę. Žmonių aptikimo algoritmas, kur naudojami HOG bruožai ir slenkantis langas, pateikiamas 1.5 pav. [13].

1.5 pav.Žmogaus aptikimas slenkančio lango metodu

Objekto išvaizda ir forma gali būti apibūdinama lokaliu intensyvumu arba krašto kryptimi, net ir neturint žinių apie šiuos parametrus. Praktiškai tai įgyvendinama dalijant vaizdo langą į mažus erdvinius regionus ("celes"), kiekvienoje celėje yra sudaroma lokali 1-D gradientų krypčių histograma. Sukombinavus histogramas gaunamas vaizdo atvaizdavimas, su išskirtais bruožais, šis bruožų išskyrimo metodas yra vadinamas orientuotų gradientų histogramos metodu (HOG) [13].

Norint surasti gradientą reikia lango I (dydis priklauso nuo celės dydžio) iš pilkos spalvos nuotraukos. Pirma surandamos gradiento pagal I sudedamosios dalys I_x ir I_y įvertinat jų centrinius skirtumus:

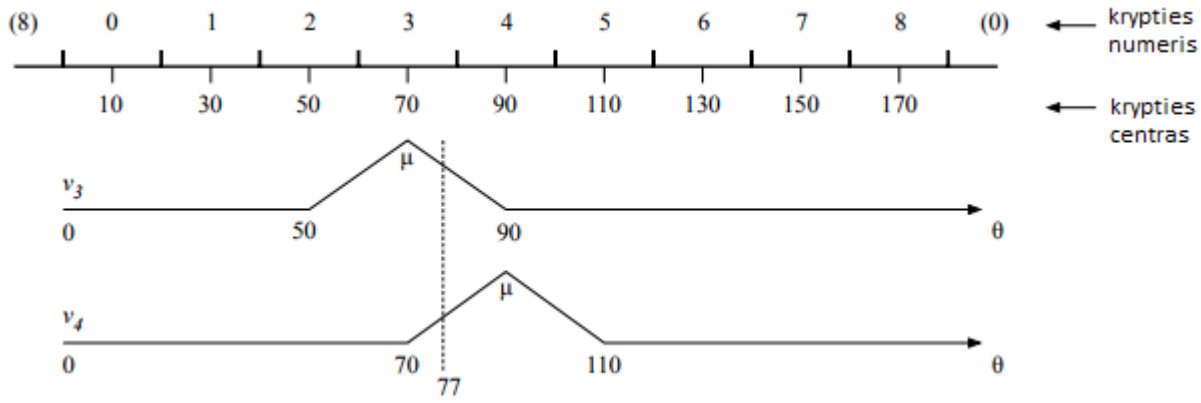
$$I_x(r,c) = I(r,c+1) - I(r,c-1)\ ir\ I_y(r,c) = I(r-1,c) - I(r+1,c). \quad (1.11)$$

Tada gradientas transformuojamas į polines koordinates, jų kampas yra ribojamas nuo 0 iki 180 laipsnių, kad gradientai, kurie rodo į skirtingas kryptis, turėtų vienodą kampą:

$$\mu = \sqrt{I_x^2 + I_y^2}\ ir\ \theta = \frac{180}{\pi}\left(tan_2^{-1}(I_y, I_x) mod\ \pi\right), \quad (1.12)$$

kur tan_2^{-1} yra keturi-ketvirčiai invertuoto tangento, kurio reikšmės svyruoja nuo – π iki π.

Langas yra padalinimas į gretimas, neperdengtas CxC pikselių dydžio celes (C=8). Kiekvienoje celėje yra suskaičiuojama orientuotų gradientų histograma B kryptimis (B=9). Kadangi yra tiek nedaug krypčių, pikselis, kurio orientacija yra artima kitos krypties ribai, gali būti priskirtas kitai krypčiai. Norint išvengti šios problemos kiekviena celė yra priskiriama dviems artimoms kryptims, maža dalis pikselio gradiento dydžio μ mažėja tiesiškai, priklausomai nuo pikselio orientacijos gradiento nutolimo nuo dviejų artimų krypčių [13].



1.6 pav.Gradiento dydžio radimas kai B=9

1.6. pav. apatinėje dalyje matoma kaip gradientai priskiriami 70 ir 90 laipsnių gretimiems krypčių centrams. Šiuo atveju gradiento orientacija yra 77 laipsniai ir gradientas yra priskiriamas 0,65 μ trečiai krypčiai ir 0,35 μ ketvirtai krypčiai. Dviejų priskyrimų suma visada lygi μ.

Celės sugrupuojamos į persidengiančius 2x2 blokus, kiekvieno bloko dydis šiuo atveju yra 2Cx2C pikselių. Du verikaliai arba horizontaliai iš eilės einantys blokai yra perdengiami dviem celėmis, tai reiškia, kad bloko žingsnis yra C pikselių. Dėl to kiekviena celė yra perdengiama keturių blokų. Keturių celių histogramos yra sujungiamos ir gaunamas vienas bruožas b kiekviename bloke ir jis yra normalizuojamas naudojant jo Euklido formą:

$$b \leftarrow \frac{b}{\sqrt{\|b\|^2+\epsilon}}, \tag{1.13}$$

kur ∈ yra nedidelė teigiama konstata, kad būtų išvengta dalybos iš nulio blokuose kuriuose nėra gradiento.

Galiausiai apskaičiuojamas HOG bruožas, sukoncentravus normalizuotų blokų bruožus į vieną vektorių (normalizavimas vykdomas du kartus, prieš ir po minimumo paieškos):

$$h \leftarrow \frac{h}{\sqrt{\|h\|^2+\epsilon}}\ , h_n \leftarrow \min(h_n, \tau), \tag{1.14}$$

čia h_n yra n-tasis h įėjimas, ir τ yra teigiamas slenkstis. Apkarpius h įėjimus, kad jie nebūtų didesni nei τ (po pirmos normalizavimo), užtikrinama, kad labai dideli gradientai neturės labai didelės įtakos, priešingu atveju kitos nuotraukos detalės nublanktų. Galutinai normalizuojant HOG bruožą, jis pasidaro nepriklausomu nuo bendro vaizdo kontrasto [13].

1.3.3.Vietiniai binariniai modeliai (LBP)

LBP modeliu paremtas bruožų išskyrimas yra paprastras, tačiau labai efektyvus. Lapeliai priskiriami pikseliams priklausomai nuo kaimyninių pikselių verčių ir parinkto slenksčio, bruožo išskyrimo rezultatas – dvejetainis skaičius. Dėl nesudėtingo skaičiavimo ir tikslaus tekstūrų išskyrimo, LPB operatorius yra labai populiarus. Didžiausias šio operatoriaus pranašumas – atsparumas monotiniškam pilkos spalvos pokyčiui, kurį sukelia nepastovus apšvietimas. Dėl nesudėtingų skaičiavimų tai vienas populiariausių metodų realaus laiko vaizdų apdorojimo aplikacijose [15].

Lokalus vietinis dvejetainis modelio operatorius, buvo pristatytas darant prielaidą, kad tekstūra turi du lokalius, papildančius vienas kitą aspektus: struktūrą ir jos intensyvumą. Bazinė LBP versija, apdoroja 3x3 pikselių bloką iškirptą iš vaizdo. Šio bloko pikseliams yra parenkama slenksčio vertė pagal centrinio pikselio vertę, šios vertės pakeliamos kvadratu ir susumuojamos, tokiu būdu gaunama centrinio pikselio vertė ir priskiriama klasė. Kadangi centrinis pikselis apsuptas 8 kaimyninių pikselių, maksimalus lapelių skaičius centriniam pikseliui 8^2=256, priskiriamas lapelis priklauso nuo kaimyninių pikselių pilkų pustonių intensyvumo [14].

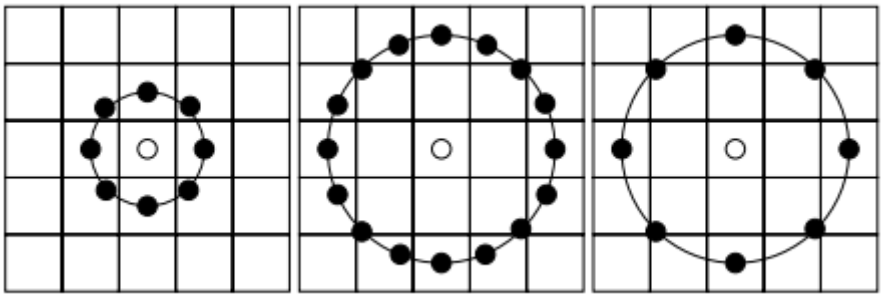
Generinė LBP formuluotė, lyginant su bazine skiriasi tuo, kad nėra limito kiek kaimyninių pikselių bus naudojama ir gali būti vertimnama ne vieno centrinio pikselio vertė. Taikymo pavyzdys, turint monochromatinį vaizdą I(x,y) ir pikselio pilko pustonio intensyvumą (x,y) nurodo parametras g_c, tada g_c=I(x,y). Laikant g_p vertinamo taško pilko pustonio intensyvumu, P - tolygiai ratu pasiskirsčiųsių pikselių skaičius, o R – spindulys aplink tašką (x,y):

$$g_p = I(x_p, y_p),\ p=0, \ldots, p-1, \tag{1.15}$$

$$x_p = x + R\cos(\frac{2\pi p}{P}), \tag{1.16}$$

$$y_p = y - R\cos(\frac{2\pi p}{P}). \tag{1.17}$$

Cirkuliarinės kaimynystės taškai vaizduojami 1.7 pav. kairiajame bloke matomas klasikinis (8,1) kaimynystės modelis, viduriniame (16,2) ir dešiniajame (8,2) modelis [14].



1.7 pav.LBP cirkuliarinė pikselių kaiminystė

Centrinio taško (x_c, y_c) LBP kodo vertė apskaičiuojama:

$$LBP_{P,R} = \sum_{p=0}^{P-1} s(g_p - g_c)2^p, \tag{1.18}$$

kur s(x) = 1, jei x>=0 ir s(x), jei x <0.

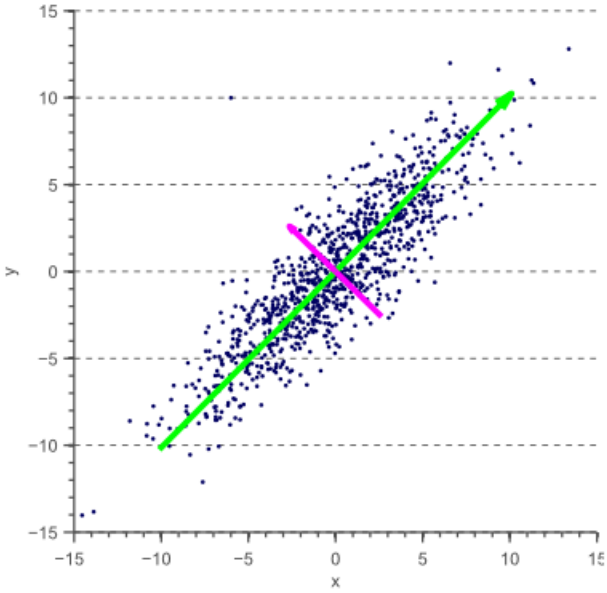
Dar vienas LBP modelio elementas yra vadinamas tolyginiu, naudojant šį metodą sumažinamas išskirtų bruožų vektoriaus ilgis. Atliekant eksperimentus naudojant LBP metodą buvo pastebėta, kad vienos binarinės sekos vaizduose atsikartoja dažniau negu kitos. Vietinis binarinis modelis laikomas tolyginiu nedaugiau bitų reikšmės pokyčių iš 0 į 1 ir atvirkščiai, kai bitų seka yra cirkuliarinė. Pavyzdžiui tolyginės sekos: 00000000 (0 perėjimų), 01100000 (2 perėjimai), netolyginės sekos: 11001001 (4 perėjimai) ir 01010010 (6 perėjimai). LBP lapeliai yra paskirstomi taip, kad visoms netolyginėms sekoms yra priskiriamas vienas lapelis, o visos tolyginėms sekos atskiri lapeliai. Kai vaizdui $f_l(x,y)$ priskiriami lapeliai, LBP histograma išreiškiama formule:

$$H_i = \sum_{x,y} I\{f_l(x,y) = i\}, i = 0, \dots, n-1, \tag{1.19}$$

kur n skirtingų LBP operatoriaus sugeneruotų lapelių skaičius ir I{A} = 1, kai A sąlyga teisinga, I{A} = 0, kai klaidinga [15].

1.3.4. Principinė komponentų analizė (PCA)

Principinė kompnonentų analizė (PCA) – linijinė transformacija paremta statistiniais metodais. Šis metodas - duomenų analitikos ir strūkturų atpažinimo įrankis, plačiai taikomas signalų ir vaizdų apdorojime, duomenų kompresijai, duomenų dimensijų sumažinimui. Šiuo metodu siekiama sumažinti duomenų dimensiją ir iš jų išgauti principines komponentes (žr. 1.7 pav.). Naudojant šią matematinę procedūrą, ieškoma krypties, kur duomenų variacija yra didžiausia [17].



1.8 pav. Principinės komponentės paieškos vektorius

Principinė komponentų analizė signalų apdorojime, tai n duotų vektorių (kintamųjų), suformuotų į K ilgio ir n-dimensijų vektorių x=[x_1, x_2, , …, x_n], transformacija į y vektorių:

$$y = A(x - m_x). \tag{1.20}$$

Svarbu turėti omenyje, kad kiekviena x vektoriaus eilė, turi K verčių priklausančių vienam įėjimui. Vektorius m_x pateiktas formulėje (1.21), tai visų įėjimų vidurkių vektorius, apibrėžtas santykiu:

$$m_x = E\{x\} = \frac{1}{K}\sum_{k=1}^{K} x_k. \tag{1.21}$$

Formulėje 1.20 matrica A yra nustatoma pagal kovariacijos matricą C_x. Matricos A eilės yra suformuotos iš savųjų vektorių (angl. *eigenvectors*) e, priklausančių matricai C_x, eilės yra išdėstomos pagal savųjų vektorių vertes, mažėjančia tvarka. Matricą C_x galima įvertinti santykiu:

$$C_x = E\{(x - m_x)(x - m_x)^T\} = \frac{1}{K}\sum_{k=1}^{K} x_k x_k^T - m_x m_x^T. \tag{1.22}$$

Įėjimo vektorius x turi n dimensijų, todėl matricos C_x dydis yra n x n. Elemntai esantys C_x{i, i} pagrindinėje įstrižainėje, reiškia x pasiskirstymą:

$$C_x(i,i) = E\{(x_i - m_i)(x_i - m_i)^T\}, \tag{1.23}$$

o kitos reikšmės C_x(i,j), tai kovariacija tarp įėjimo kinatmųjų x_i, x_j:

$$C_x(i,j) = E\left\{(x_i - m_i)(x_j - m_j)^T\right\}. \quad (1.24)$$

Matricos A eilės išreikštos 1.20 formulėje ir normalizuotos, taigi PCA inversiją galima išreikšti santykiu:

$$x = A^T y + m_x. \quad (1.25)$$

PCA brožų išskyrimo sparta, bei tikslumas priklauso nuo naudojamų treniravimo duomenų kiekio ir principinių komponenčių skaičiaus. Šis metodas neatsparus aplinkos apšvietimo pokyčiui, nes šablonai generuojami esant momentiniam apšvietimui. Dėl gana nesudėtingos realizacijos ir nedidelio skaičiavimo resursų poreikio, PCA veikimo sparta yra gana didelė [16].

1.4. Duomenų klasifikavimo metodai

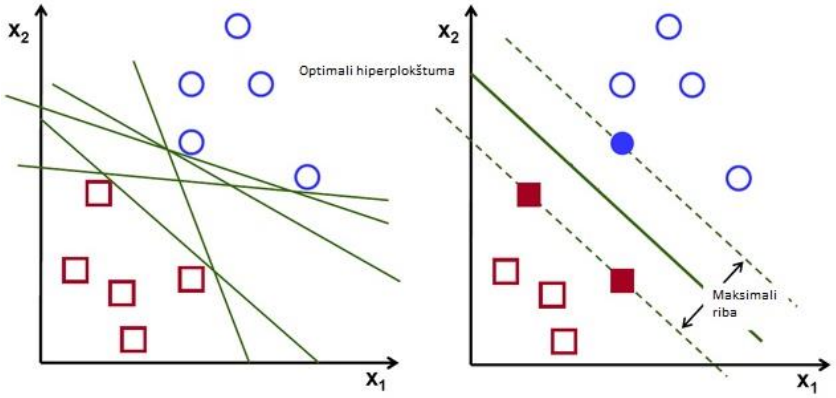
Objektų klasifikacija yra sudėtinga užduotis mašinoms. Klasifikacijos sistemoje yra duomenų bazė, kurioje yra iš anksto nustatyti objektų modeliai. Objektai priskiriami tam tikrai kategorijai, modelius lyginant su objektais esančiais vaizde. Klasifikatorius su mokytoju – tai, kai dalis pikselių yra žinomi, sugrupuoti ir jiems priskiriami atitinkami žymekliai. Šis procesas vadinamas klasifikatoriaus mokymu. Vėliau klasifikatorius išmoktus pikselius naudoja kitų vaizdų klasifikacijai. Klasifikatorius be mokytojo – pikseliai yra sugrupuojami, pagal skirtingus jų parametrus. Šis procesas vadinamas grupavimu. Vartotojas parenka reikiamą pikselių grupių skaičių. Šis metodas naudojamas tada, kai nėra galimybės naudoti apmokytus pikselius. Toliau bus apžvelgiami pagrindiniai klasifikacijos metodai [8].

1.4.1. Atraminio vektoriaus mašina (SVM)

Atraminio vektoriaus mašina (SVM) – klasifikatoriaus su mokytoju tipas, kurio veikimo principas pagrįstas sprendimų ribos suradimu. SVM apibrėžia hiperplokštumą priklausomai nuo apmokymo duomenų. Hiperplokštuma suformuoja sprendimų ribas, pagal kurias atliekama klasifikacija. Hiperplokštuma yra suprojektuota taip, kad galėtų skirstyti įėjimo duomenis į dvi klases, pagal sudarytą sprendimų funkciją. Dauguma SVM klasifikatorių gali skirstyti objektus į dvi klases, tačiau esant poreikiui galima naudoti ir daugiapakopį klasifikatorių.

Hiperplokštumą galima apibūdinti, kaip tiesę, aprašytą tam tikra funkcija. Iš visų sudarytų tiesių, reikia surasti optimaliausią funkciją tinkančią visiems duomenų taškams. Tokia tiesė parenkama, pagal didžiausią nuotolį nuo visų duomenų taškų, tokiu būdu sumažinama triukšmo

įtaka. SVM algoritmo tikslas yra surasti hiperplokštumą, kuri apibrėžia didžiausią minimalų atstumą iki treniravimo pavyzdžių. SVM teorijoje šis atstumas vadinamas duomenų riba [10]. Optimali skiriamoji hiperplokštuma suranda duomenų ribos maksimumą 1.9 pav.



1.9 pav.Optimalios hiperplokštumos suradimas

Kai SVM klasifikatoriui paduodami dviejų klasių treniravimo vektoriai $x_i \in R^d$, i= 1, …, l ir lapelių vektorius $y \in \{1, -1\}^l$, taigi SVM turi išspręsti šią optimizavimo problemą:

$$\begin{gathered} \min_{w\in H, b\in R, \varepsilon_i \in R} \frac{1}{2} w^T w + C \sum_{i=1}^{l} \varepsilon_i , \\ priklauso\ nuo\ \ y_i(w^T \varphi(x_i) + b) \geq 1 - \varepsilon_i, \\ \varepsilon_i \geq 0, \ \ i = 1, \dots, l, \end{gathered} \tag{1.26}$$

kur $w \in R^d$ – svorių vektorius $c \in R_+$ sureguliavimo konstanta ir funkcija φ projektuoja treniravimo duomenis į atitinkama duomenų plokštumą H, tokiu būdu gaunami netiesiniai sprendimų paviršiai. SVM metodu šios probelmos sprendimas išreiškiamas formule:

$$\begin{gathered} \min_{\alpha \in R^l} \frac{1}{2} \alpha^T (Ko(yy^T))\alpha - e^T \alpha, \\ \text{priklauso nuo } 0 < \alpha \leq Ce, y^T \alpha = 0, \end{gathered} \tag{1.27}$$

kur $e \in R^l$ visų loginių 1 vektorius, $K \in R^{lxl}$ branduolio (angl. *Kernel*) matrica, o žymi *Hadamard–Schur* produktą [18].

Reikia atpažinti kelis teisėjo gestus, todėl bus reikalingas kelių klasių SVM klasifikatorius. Yra du pagrindiniai kelių klasių atraminio vektoriaus mašinos klasifikavimo metodai: vienas-prieš-vieną arba vienas-prieš-visus. Vienas-prieš-visus (1VR) sudaro k atskirų binarinių klasifikatorių, atitinkamam k-klasių skaičiui. M-tasis binarinis klasifikatorius treniruojamas naudojant duomenis iš m-tosios klasės, laikant juos teigiamais duomenimis, o likusios k-1 klasės laikomos neigiamomis. Testavimo metu, klasės žymeklis nustatomas naudojant binarinį klasifikatorių, kuris išduodą maksimalią klasifikavimo vertę. Nesubalansuotas treniravimo duomenų kiekis daro didelę įtaką 1VR SVM klasifikatoriui. Labai svarbu, kad kiekvienos klasės treniravimo duomenų kiekis būtų vienodas, tokiu atveju santykis tarp teigiamų ir neigiamų

pavyzdžių kiekviename klasifikatoriuje yra $\frac{1}{k-1}$. Tokiu atveju prarandama pirminės problemos simetrija [18].

Vienas-prieš-vieną metodas – vertina visus įmanomus porinius klasifikatorius, taigi galutinis invidualių binarinių klasifikatorių skaičius būtų k(k-1)/2. Pritaikius visus klasifikatorius testavimo duomenims, kiekvienas klasifikatorius duotų vieną balsą didžiausią tikimybę turinčiai klasei. Prognuozuojama ta klasė, kuri surenka daugiausiai balsų. Šie klasifikatoriai yra daug didesni lyginant su vienas-prieš-visus metodu, tačiau kiekvienam binariniam klasifikatoriui reikalingi skaičiavimo resursai yra mažesni, todėl šiuo metodu realizuotas klasifikatorius yra spartesnis [18].

1.4.2. Sprendimų medžiai

Sprendimų medis – ne parametrų klasifikatorius, tai mašinų mokymosi algoritmas paremtas pavyzdžiais. Sprendimų medis padalija įėjimo duomenis į atskirus elementus, kurie priskiriami vienai klasei. Padalijimas atvaizduojamas, kaip testų seka, sudaryta iš šakninio mazgo, vidinių mazgų rinkinio ir išvadinių mazgų vadinamų „lapais“. Šakniniai ir vidiniai mazgai kolektyviai laikomi neišvadiniais mazgais, suskirstytais į sprendimų etapus. Kiekvienas vidinis mazgas, turi įėjimo duomenų vertę ir atšakose iš mazgų nurodomi galimi išėjimo rezultatai [11]. Galima naudoti ir daugiau negu vieną sprendimų medį klasifikuojant, toks klasifikatorius vadinamas „Atsitiktiniai miškai“ (angl. *Random Forests*).

Dauguma sprendimų medžių algoritmų turi dvi skirtingas fazes: statymo arba auginimo fazė, po kurios seka genėjimo (angl. *pruning*) fazė. Medžio statymo etape treniravimo duomenys yra rekursyviai suskaidomi, kol visi įrašai priskiriami vienai klasei. Kiekvienam padalijimui (angl. *partition*), prie sprendimų medžio pridedamas naujas mazgas, iš pradžių, medis turi tik vieną šakninį mazgą visai treniravimo duomenų bazei. Įrašų rinkiniui esančiam viename padalijime P, nustatomas testų kriterijus T, tada atliekamas tolimesnis medžio dalijimas P_1 ... P_m. Naujos šakos yra sukuriamos visiems padalijimams P_1 ... P_m, šios šakos yra prikiriamos prie P kaip jo vaikai. Šakai, kuri priklauso P, suteikiamas lapelis su testo duomenimis T, tada P_1 ... P_m yra toliau rekursyviai dalijamos. Particija, kurioje visos klasės reikšmės yra vienodos, toliau nebėra dalinama ir „lapas“ susijęs su šia particija išduodamas kaip rezultatas – klasė [19].

Siekiant išvengti persipildymo (angl. *overfitting*) reikalingas sprendimų medžio genėjimas, kuris generalizuoja medžio struktūrą. Geriausias sprendimų medžio modelis, ne tik tiksliai klasifikuoja duomenis, bet ir turi mažiausią įmanomą skaičių užkoduotų bitų. Genėjimo fazės

tikslas, surasti medį medyje (angl. *subtree*), kuriame būtų nurodytas optimiliausias kelias iki sprendimo šakos [19].

Daugialypiame sprendimų medyje mazgų t skėlimo tvarka, visiem prognuozuojamiems veiksniams (angl. *predictors*) x_i, i=1, …, p:

1. Mazgams t, i_t sprendimų medis suskaičiuoja svertinius priemaišus;
2. Tkrinama tikimybė, kad sprendimas yra tikrinamame mazge t:

$$P(T) = \sum_{j \in T} w_j, \tag{1.28}$$

kur w_j sprendimo j svoris, o T – visi sprendimai esantys mazge t.

3. Sprendimų medis surūšiuoja x_i didėjančia tvarka. Kiekviename mazge surūšiuoti kandidatai gali būti padalijimo veiksniai.
4. Sprendimų medyje nusprendžiama kuria šaka turėtų keliauti sprendimas x_i iš mazgo t, maksimaliai padidinant neskaidrumo stiprį (ΔI) tarp visų skėlimo kandidatų. Visiems skėlimo kandidatams x_i:
    - Kiekveiname mazge t tikimybė skeliama į kairį ir dešinį vaikų mazgus (atitinkamai t_L ir t_R).
    - Sprendimų medis suskaičiuoja ΔI. Jeigu x_i netrūksta jokių verčių, tada neskaidrumo koeficientas išreiškiamas formule:

$$\Delta I = P(t)i_t - P(T_L)i_{t_L} - P(T_R)i_{T_R}. \tag{1.29}$$

    - Jeigu x_i trūksta verčių, laiko kad stebėjimų (angl. *observations*), trūksta atsitiktine tvarka ir neskaidrumo koeficientas apskaičiuojamas:

$$\Delta I_U = P(T - T_U)i_t - P(T_L)i_{t_L} - P(T_R)i_{T_R}, \tag{1.30}$$

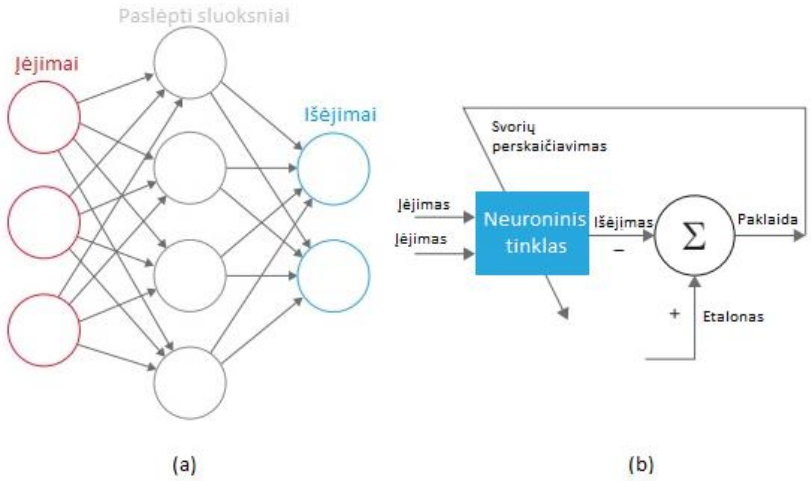
kur $T\text{-}T_U$ visų stėbejimų skaičius tam tikrame mazge t [20].

1.4.3. Dirbtiniai neuroniai tinklai (ANN)

Dirbtinis neuroninis tinklas (ANN) – tarpusavyje sujungtų, dirbtinių neuronų, kurie tarpusavyje keičiasi pranešimais sistema. Sujungimai tarp neuronų turi skaitinius svorius, neuroninio tinklo apmokymo metu, šie svoriai yra koreguojami, kad būtų išvedamas atpažintas objektas ar modelis. Tinklas yra sudarytas iš kelių sluoksnių, bruožus – aptinkančių „neuronų“. Kiekviename sluoksnyje yra daug neuronų, kurie reaguoja į skirtingas įėjimų kombinacijas iš ankstesnių sluoksnių. Paveikslėlyje 1.10 (a) vaizduojama dirbtinio neuroninio tinklo struktūra.

Neuroninio tinklo sluoksniai sudaryti taip, kad pirmajame sluoksnyje aptinkama grupė primityvių modelių iš įėjimo duomenų, antrame sluoksnyje aptinkamos šių modelių struktūros,

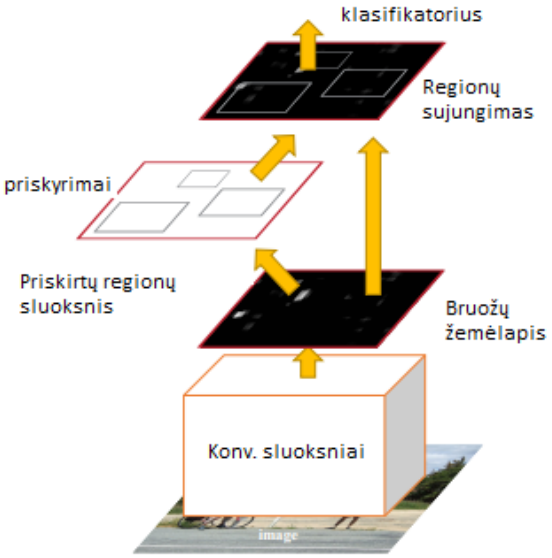
trečiame sluoksnyje aptinkamos struktūros iš ankstesnio sluoksnio modelio ir t.t. Tipiniame dirbtiniame neuroniniame tinkle yra 5 – 25, atskiri sluoksniai. Mokymas atliekamas įėjimuose naudojant daug sugrupuotų duomenų rinkinių, kuriems priskiriami numatomi išėjimo žymėjimai. Apmokymui naudojami bendrosios paskirties metodai, kad būtų nustatyti tarpinių ir galutinių neuronų svoriai. Blokinė neuroninio tinklo apmokymo procedūra vaizduojama 1.10 (b) pav. [9].



1.10 pav.Dirbtinis neuroninis tinklas (a) ir neuroninio tinklo apmokymas (b)

Kadangi darbe reikia atpažinti objektus, objektai tai teisėjas rodantis gestą. Pastaraisiais metais geriausi rezultai šioje srityje, pasiekti naudojant konvoliucinius neuroninius tinklus (CNN), tiksliau regiono pagrindu paremti konvoliuciniai neuroniai tinklai (R-CNN). Tokios architektūros neuoriniame tinkle, bruožų išskyrimui naudojami paslėpti neuroniai sluoksniai. Regiono teikimo (angl. *region proposal*) apmokymo metodas parpastai pasikliauja nesudėtingais objekto (angl. *inexpensive features*) bruožais. Selektyvinė paieška (angl. *selective search*) – populiariausias bruožų išskyrimo metodas gilaus mokymosi (angl. *deep learning*) srityje. Priklausomai nuo suprojektuotų žemo lygio bruožų, godžiai sujungia super pikselius (angl. *superpixels*), ryškiausius regiono bruožus. Lygininat su kitais klasifikavimo algoritmais, selektyvi paieška, yra gerokai lėtesnė, dirbant su vienu centriniu abdorojimo bloku (CPU), vienam paveiksliukui bruožų išskyrimas užtrunka apie 2 sekundes.

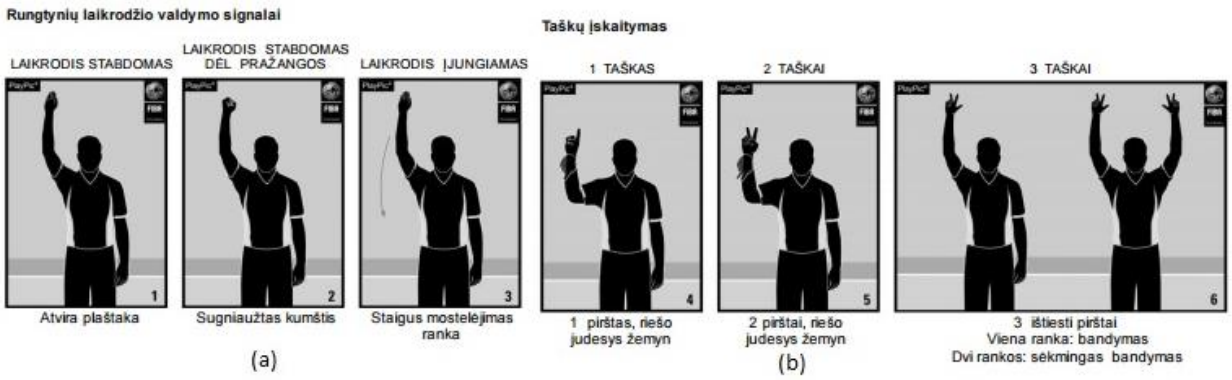
Nors regiono pagrindu paremti detektoriai, gali rasti obkjektą įvairios rezoliucijos nuotraukose, ankščiau apžvelgtu, paveikslų piramidės metodu, tačiau geresnė greitaveika, kai naudojamos vienos rezoliucijos nuotraukos. Gilaus mokymosi algoritmu, paremtas objektų atpažinimas, tai juodosios dežės modelis. Pilnai sujungtas neuronų sluoksnis (angl. *fully connected layer*), naudojamas atrasti lango koordinates, kuriame yra norimas atpažinti objektas. Pilnai sujungtas neuronų sluoksnis, transformuojamas į konvoliucinį sluoksnį, kad būtų aptikti kelių klasių objektai [21]. Šis modelis pateikiamas 1.11 pav.



1.11 pav.R-CNN klasifikatoriaus architektūra

1.5. Krepšinio teisėjo gestai

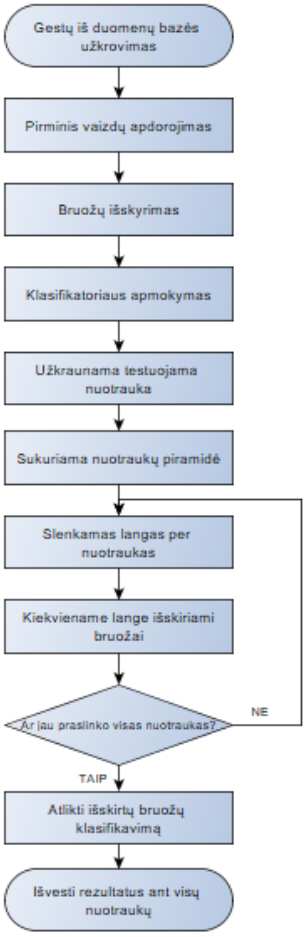
Sklandžią krepšinio varžybų eigą ir tvarką aikštelėje užtikrina krepšinio teisėjai. Paprastai jų būna trys – vienas stebi veiksmą aikštės viduryje, kiti du iš kraštų. Teisėjai komandas rodo varžybų sekretoriatui, kuris atitinkamai valdo varžybų laiko stabdymą ar startą, bei pagal varžybų protokolą priskiria pražangas žaidėjams ir jos yra indukuojamos švieslentėje. Komunikacija tarp varžybų sekretoriato ir teisėjų yra labai svarbi, ji realizuojama naudojantis standartiniais teisėjo gestais. Teisėjo komandos skirstomos į tokias grupes: rungtynių laikrodžio valdymo signalai (1.12 pav. a), taškų įskaitymas (1.12 pav. b), žaidėjų keitimas ir minutės pertraukėlė, informatyvūs gestai, pažeidimai, žaidėjų numeriai, pražangų pobūdis, baudų už pražangas vykdymas, signalai rodomi sekretoriatui, baudos metimų vykdymas [7].



1.12 pav.Rungtynių laikrodžio valdymo signalai (a) ir taškų įskaitymas (b)

2. Metodika ir sistemos realizacija

Krepšinio teisėjo gestų atpažinimo sistema yra objektų atpažinimo sistema, atpažįstami objektai yra teisėjo rodomi gestai. Sistema buvo sukurta „MATLAB“ aplinkoje. R-CNN klasifikatoriui buvo naudojama integruota funkcija, o atraminio vektoriaus mašinos (SVM) ir sprendimo medžių (RF) gestų aptikimo bendras sistemos algoritmas vaizduojamas 2.1 pav. Užkrovus treniravimo duomenų bazę, atliekamas visų joje esančių nuotraukų pirminis apdorojimas, vėliau pasirinktu metodu iš kiekvienos nuotraukos išskiriami bruožai ir atliekamas klasifikatorių apmokymas. Įkėlus nuotrauką, dydis pakeičiamas į kelias skirtingas rezoliucijas ir per kiekvieną nuotrauką slenkamas langas. Kiekviename lange išskirami bruožai ir atliekama visų langų klasifikacija, rezultatas išvedamas ant ekrano.



2.1 pav. Gestų aptikimo sistemos bendras veikimo algoritmas

Sitemos kūrimo etapą sudaro tokie žingsniai:

1. Duomenų bazės paruošimas, video medžiagos karpymas, gestų iškirpimas;
2. Pirminis treniravimo duomenų apdorojimas;

3. Nuotraukų pavertimas į požymių vektorius;
4. Klasifikatoriaus apmokymas;
5. Slenkančio lango funkcijos sukūrimas.

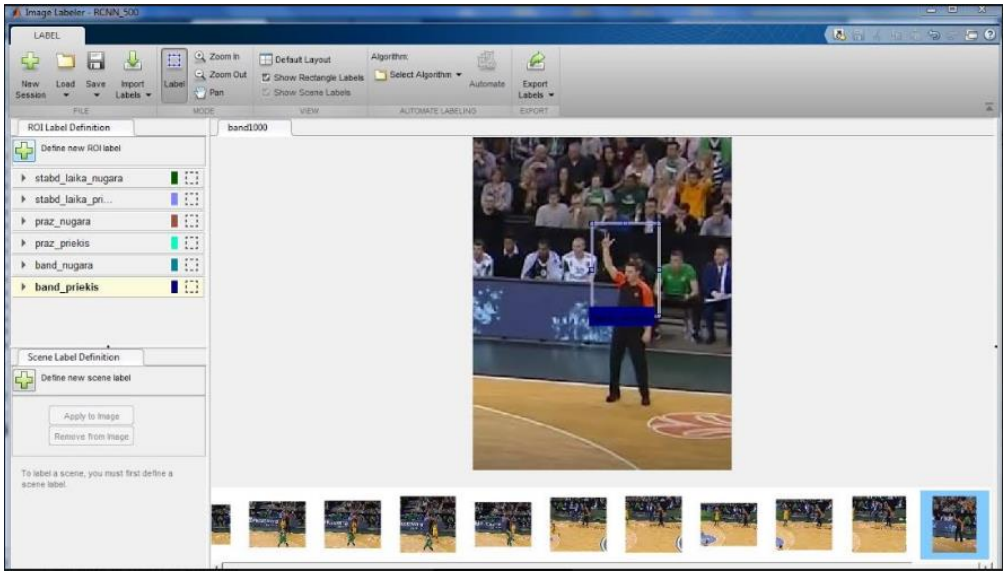
2.1. Gestų duomenų bazės sudarymas

Buvo nuspręsta sukurti sistemą, kuri atpažįsta statines teisėjo rodomas komandas. Analizuojant video medžiagą, buvo pastebėta, kad dažniausiai rodomi teisėjo gestai yra: trys taškai – bandymas, stabdyti laiką ir stabdyti laiką pražanga. Šie gestai ir bus atpažįstami sistemoje, pavyzdžiai iš surinktos duomenų bazės parodyti 2.2 pav. Kadangi klasifikatoriams svarbi objekto orientacija kameros atžvilgiu, buvo nuspręsta atskiroms klasėms priskirti kai teisėjas stovi priešais kamerą ir kai nusisukęs nuo jos, taigi iš viso gautos 6 klasės ir fonas. Iš viso iš video medžiagos buvo iškirpta 3000 nuotraukų, kur teisėjas rodo klasei priskiriamą gestą. Prasukinėjant video medžiagą po kadrą buvo renkama kuo įvairesnė mokymosi medžiagą (ranka pkeitusi poziciją, rotaciją, fono pasikeitimai ir kt.). Teisėjo gestai buvo karpomi iš „Eurolygos“ 2017-2018 sezono varžybų įrašų.

2.2 pav. Gestų duomenų bazės pavyzdžiai

R-CNN objektų detektoriaus duomenų paruošimui, buvo naudojama *ImageLabeler* aplikacija, kur pažymimas nuotraukos regionas (ROI), kuri bus naudojamas kaip klasifikatoriaus duomenys (2.3 pav). Visa kita informacija nuotraukoje yra laikoma fonu. „MATLAB“ aplinkoje

sukuriama lentelė, kurioje nurodoma nuotraukos vieta kompiuteryje, bei lango koordinatės ir jo dydis.

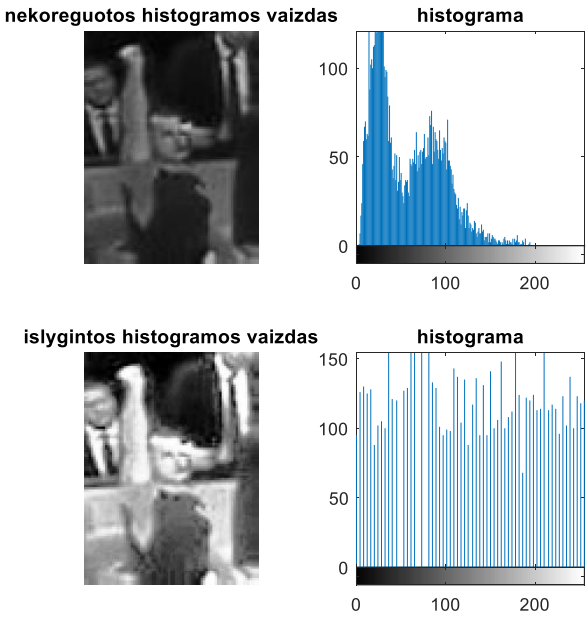


2.3 pav. Dominančių regionų priskyrimas su *ImageLabeler* aplikacija

2.2. Pirminis duomenų apdorojimas

Pirminio apdorojimo veiksmai buvo atliekami „MATLAB“ programiniu paketu. Treniravimo duomenų nuotraukos sudarytos iš RGB splavų paletės. Dirbant su vaizdų apdorojimu šią spalvų paletę reikia konvertuoti į kitą, teisėjo gestų atpažinimo sistemoje, splavų palėtė buvo konvertuota į juoda/balta su pilkais atspalviais, naudojant funkciją *rgb2gray*. Vėliau buvo bandomos įvairios kitos išankstinio apdorojimo kombinacijos: kraštų aptikimas *Prewitt* ir *Sobel* metodais, vaizdo binarizavimas, odos spalvos ir marškinėlių spalvos segmentavimas. Visi klasifikatoriai po šių pirminio apdorojimo žingsnių parodė prastesnius rezultatus. Binarizuoti treniravimo duomenys yra netikslūs ir aiškiai neišskiria dominančio objekto – teisėjo iš fono, nes fone taip pat yra ryškių objektų, kurie užgožia teisėją ir matomas vaizdas panašus į triukšmą, aiškios gestų formos neišskiriamos.

Remiantis išanalizuotuose šaltiniuose pateikta informacija, histogramos išlyginimas padidina vaizdo kontrastą, naudojant šį pirminio apdorojimo metodą, klasifikavimo rezultatai pagerėjo. Nors paryškinamas ne tik teisėjas, bet ir fonas, tačiau po išlyginimo geriau matomas teisėjo rodomas gestas. „MATLAB“ aplinkoje histogramos išlyginimas realziuotas su *histeq* funkcija, kuri parenka kurią histogramos dalį minimizuoti (2.4 pav).



2.4 pav. Vaizdo histogramos išlyginimas

2.3. Bruožų išskyrimas

Bruožų išskyrimui naudojami trys metodai: LBP, HOG ir PCA. R-CNN nereikalingas bruožų išskyrimas, jie išskiriami paslėptuose sluoksniuose. Kiekviename paveiksle iš duomenų bazės ir kiekviename slenkančio lango regione atliekamas bruožų išskyrimas. Bruožai išskiriami iš sukurto *ImageDatastore* objekto.

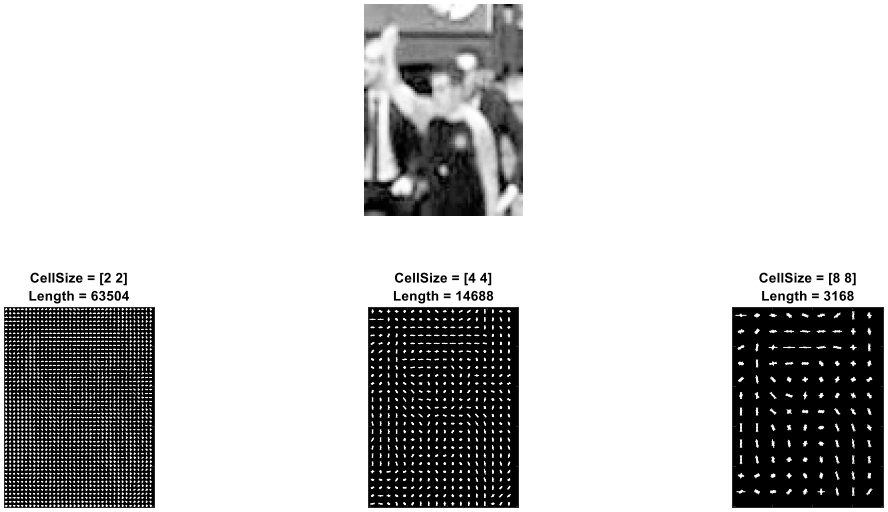
Bruožų išskyrimas HOG metodu

HOG bruožų išskyrimas „MATLAB“ aplinkoje vykdomas komanda *extractHOGFeatures*. Gaunamas HOG bruožas, sudarytas iš celių histogramų, kurių yra keturis kartus daugiau nei blokų. Šiame darbe vienos iš naudojamų treniravimo nuotraukos rezoliucija 100x75. Pavyzdžiui jeigu bus naudojama 4x4 pikselių celė, tada nuotraukoje tilps 18 celių horizontaliai ir 25 celės vertikaliai. Žingsnis šiuo atveju yra keturi pikseliai, todėl gaunamas 24 blokas vertikaliai ir 17 blokų horizontaliai, kadangi bloką sudaro keturios celės ir kiekvienai histogramai priskiriamos devynios kryptys, tada h vektoriaus gaunamas ilgis yra:

$$h = 24 * 17 * 4 * 9 = 14688. \quad (2.1)$$

Bruožų išskyrimo funkcijoje konfigūruojami parametrai: celės dydis, bloko dydis, reguliuojamas blokų persidengimas, orientuotų histogramų skaičius. Naudojant skirtingas šių

parametrų kombinacijas galima gauti geresnį rezultatą klasifikavimo etape, tačiau ilgesnis bruožų vektorius lemia klasifikavimo spartos sumažėjimą. HOG bruožų vizualizacija su skirtingais celių dydžiais pateikta 2.5 pav.



2.5 pav. Orientuotų gradientų histogramos vizualizacija su skirtingai celės dydžiais

LBP bruožų išskyrimas

LBP bruožų vektoriaus išskyrimas iš vaizdo realizuojamas funkcija *extractLBPFeatures*. Funkcijos LBP bruožų vektoriaus ilgis yra 1-N, kur N yra bruožų skaičius. Visų bruožų skaitinės reikšmės svyruoja nuo 0 iki 1. LBP bruožai koduoja informaciją apie lokalią tekstūrą. Funkcija padalina nuotrauką į nepersidengiančias celes. Norint surinkti informaciją iš didesnių nuotraukos regionų, reikia didinti celės dimensijas, tačiau tada prarandamos smulkesnės nuotraukos detalės. N priklauso nuo celių skaičiaus nuotraukoje – n, pasirinkto kaimynų skaičiaus – P ir stataus (angl. *upright*) bruožų išskyrimo parametro. *Upright* tai rotacijos nekeičiamumo (angl. *rotation invariance*) vėliava, ją pasirinkus bruožų vektoriuje nėra užkoduota nuotraukos (dominančio objekto) rotacijos informacija.

LBP bruožų vektoriaus ilgis išreikštas formulėje 2.2. Kiekvienos celės dydis priklauso nuo dėžių (angl. *bins*) B skaičiaus. Kai rotacija yra nekeičiama dėžių skaičius išreiškiamas B=(Px(P-1)+3, kur P – kaimynų skaičius. Pavyzdžiui, turint 100x75 pikselių nuotrauką, visoje nuotraukoje tilps 450 celių, vertinamų kaimynų skaičius – 8. Tada dėžių skaičius bus B=(8x(8-1)+3=59, tada galutinis vektoriaus ilgis apskaičiuojamas:

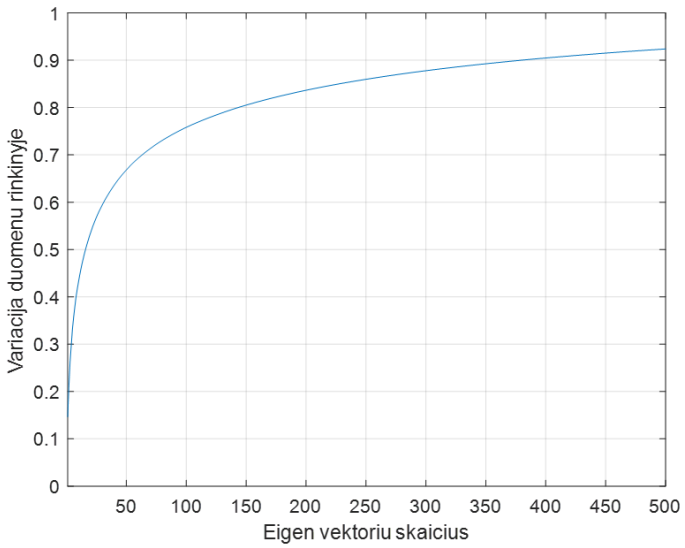
$$N = numCells * B = 450 * 59 = 26550. \quad (2.2)$$

Pagrindinė komponentų analizė (PCA)

PCA metodas paremtas – tarpusavyje koreliuojančių kintamųjų transformacija į mažesnį kiekį, tarpusavyje nekoreliuojančių, kintamųjų. Iš duomenų masyvo pagrindiniai PCA vektoriai išmokomi be mokytojo. PCA gali būti pritaikytas gestų atpažinimo užduočiai – vaizdo pikseliai paverčiami į tam tikrą kiekį savųjų vektorių (angl. *eigenvectors*), kurie gali būti naudojami palyginti panašumamas tarp kelių nuotraukų. PCA veikia geriausiai, kai objekto rotacija treniravimo duomenų nuotraukose keičiasi nežymiai. Prieš pradedant savųjų vektorių skaičiavimus, visi treniravimo vaizdai konvertuojami į pilką spalvą ir išlyginamos jų histogramos. Tada visi vaizdai konvertuojami į skaitines matricas, su funkcija *double*. Matricos dimensijos nxm, kur n yra nuotraukos pikselių skaičius, o m nuotraukų skaičius. Nuotraukų konvertavimo į savuosius vektoriu žingsniai:

1) Apskaičiuojami įvesties vaizdų (konvertuotų į mxn matricas) vidurkiai – su *mean* funkcija;
2) Iš kiekvienos pikselio vertės atimamas gautas vidurkis, tokiu būdu gaunami vidutinio poslinkio vaizdai (angl. *mean shifted*);
3) Suskaičiuojami savieji vektoriai ir savybės (angl. *eigenvalues*);
4) Surūšiuojami savieji vektoriai pagal jų savybes, mažėjančia tvarka;
5) Paliekami tie savieji vektoriai, kurių savybių vertės turi didžiausias vertes (paliekamos aktualiausios principinės komponentės);
6) Savieji vektoriai suprojektuojami į mxn matricą, kur m – pasirinktas kiekis svarbiausių savųjų vektorių reikšmių, n – nuotraukų kiekis.

Pavyzdžiui, turime 3500 nuotraukų, kurių dydis 100x75. Konvertavus į skaitinę mxn matricą bus gaunama m=100x75=7500 (piksellių skaičius), n = 3500 (nuotraukų skaičius). Apskaičiavus matricos vidurkį gaunama 1x7500 matrica, kur pirmame stulpelyje yra visų 3500 pikselių vidurkiai. Savieji vektoriai ir savybės suskaičiuojamos iš vaizdų matricos, naudojant „MATLAB“ funkciją *pca*. Su šia funkcija atliekamas ir reikšmių surūšiavimas (4 žingsnis). Tada pasirenkami svarbiausi savieji vektoriai. Reikšmingi savieji vektoriai – didžiausią procentinę dalį duomenų variacijos apibrėžiantys savieji vektoriai t.y. nurodo, kiek reikia naudoti principinių komponenčių norint korektiškai suklasifikuoti duomenis. Tyrimo metu nustatyta, kad duomenys klasifikuojami tinkamai, kai variacija tarp jų yra apie 90 procentų, tai atitinka 500 pirmų principinių komponenčių (2.6 pav.).



2.6 pav. Optimalus savųjų vektorių skaičius duomenų klasifikacijai

2.4. Klasifikatoriai

Duomenų klasifikacijai naudojamos funkcijos iš „MATLAB“ išplėtimų paketo, „Image processing toolbox“. SVM ir atsitiktinių miškų klasifikatoriams, treniravimo duomenys turi būti pateikti kaip *ImageDatastore* objektas, kur nurodomas paveikslų dydis, skaičius, klasių vardai ir t.t. R-CNN objektų aptikimo algoritmas apmokomas funkcija trainFasterRCNNObjectDetector, jam reikalingos nuotraukose esantys dominantys regionai (angl. *regions of interests*), kurie sukuriami su *ImageLabeler* plėtiniu.

Daugialypis SVM klasifikatorius

Daugialypis SVM klasifikatorius „MATLAB“ aplinkoje aprašomas funkcijoje *fitcecoc*. Daugialypiame SVM kalsifikatoriuje parenkama po vieną optimalią hiperplokštumą kiekvienai duomenų klasei. *Fitcecoc* funkcija gražina ištreniruotą, daugialypį, išvesties kodų klaidas taisantį modelį (angl. *Error-correcting output codes* (ECOC) model). Funkcija naudoja K(K-1)/2 binarinę atraminio vektoriaus mašiną, kuri naudoja vienas-prieš-vieną (angl. *one-vs-one*) kodavimo dizainą, kur K – klasių skaičius.

ECOC klasifikacijai reikalingas kodavimo dizainas, nustatantis klases, reikalingas binaraninių klasifikatorių treniravimui ir dekodavimo schema, kurioje nustatomi apibebndrinti binarinio klasifikatoriaus rezultatai. Sakykime, kad: turime tris klases, kodavimo dizainas –

vienas-prieš-vieną, dekodavimo schema naudoja praradimą (angl. *loss*) g. Klasifikavimo modelio sudarymui, ECOC vykdomi šie žingsniai:

1) Vienas-prieš-vieną kodavimo dizainas, kai turim tris klases:

	Mokinys 1	Mokinys 2	Mokinys 3
Klasė 1	1	1	0
Klasė 2	-1	0	1
Klasė 3	0	-1	-1

2) Mokinys mokosi iš pirmos ir antros klasių, atitinkamai laikydamas klasę 1 teigiamu, o antrą klasę neigiamu pavyzdžiu. Kiti mokiniai apmokomi panašiu metodu. M laikome kodavimo dizaino matrica su elementais m_{kl}, o s_l kalsifikavimo taškai teigiamai mokinio klasei l.

3) Naujas pastebėjimas (angl. *observation*) yra priskiriamas klasei (k), kuri minimizuoja binarinių praradimų skaičių:

$$k = \underset{k}{\operatorname{argmin}} \frac{\sum_{l=1}^{L} |m_{kl}| g(m_{kl}, s_l)}{\sum_{l=1}^{L} |m_{kl}|}. \quad (2.3)$$

Kodavimo dizaino matricos elementai nurodo, kurios klasės naudojamos binarinio mokinio mokymui, daugialypė problema sumažinama iki dvejetainės problemos. Kiekviena matricos eilutė yra klasė, o stulpelis – mokinys. Kiekviena eilutė lygi 1, nurodo, kad bus klasifikuojama, kaip teigiama klasė, kai eilutė -1, klasifikuojama, kaip neigiama klasė, o kai 0 ignoruojama. Visi galimi kodavimo dizainai „MATLAB“ aplinkoje pateikti 2.1 lentelėje.

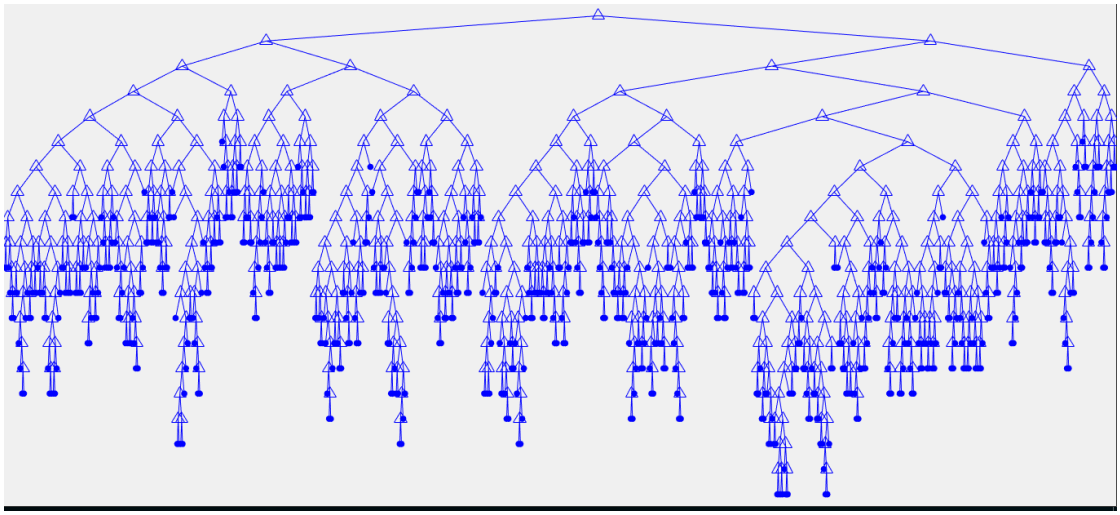
2.1 lentelė ECOC kodavimo dizainai „MATLAB“ aplinkoje

Kodavimo dizainas	Apibūdinimas	Mokinių skaičius
Vienas-prieš-visus	Kiekvienam binariniam mokiniui, viena klasė teigiama, likusios neigiamos.	K
Vienas-prieš-vieną	Kiekvienam binariniam mokiniui, viena klasė teigiama kita neigiama, o likusios ignoruojamos.	K(K-1)/2
Užbaigtas binarinis	Šis diziainas panašus į vienas-prieš-vieną, tik visos klasės yra vertinamos t.y. jom priskiriamos 1 arba -1 vertės.	2^{K-1}-1
Užbaigtas tris kartus	Kiekvienai klasei taikomos visos vertės, -1, 1 ir 0, yra trys klasifikavimo iteracijos, kiekvienoje iteracijoje bent po vieną klasę reikia priskirti -1 ir	$(3^K-2^{K+1}+1)/2$

	1.	
Kvadratinis	Pirmas binarinis mokinys priskiria pirmą klasę 1, o vsas kitas laiko -1, antras binarinis mokinys antrą klasę priskiria 1, o visas kitas laiko 0 ir t.t.	K-1
Atsitiktinai sutankintas	Kiekvienam binariniam mokiniui, atsitiktinai priskiriamos klasių vertės, bent po vieną kiekvieno tipo.	Atsitiktinis, bet maždaug $10\log_2 K$
Atsitiktinai praretintas	Taip pat atsitiktinai priskiria klasių vertes, su 0,25 tikimybe kiekvienai, klasės su mažesne nei 0,5 tikimybe yra ignoruojamos.	Atsitiktinis, bet maždaug $15\log_2 K$

Sprendimų medžių miškas

Sprendimų medžių miškas kitaip vadinamas, atsitiktiniu miškų (angl. *random forests – RF*) klasifikatoriumi. Atsitiktinių miškų klasifikavimo algoritmas, „MATLAB“ aplinkoje realizuojamas su funkcija *TreeBagger*. Su šia funkcija sukuriamas sprendimų medžių masyvas. Kiekviename medyje priklausomai nuo treniravimo duomenų gautų iš bruožų vektorių, priimamas sprendimos kokiai klasei priskirti bruožų vektorių. Visų medžių balsai (angl. *votes*) už kalses patikrinami ir ta klasė, kuri surenka daugiausiai balsų ir yra klasifikatoriaus išėjimas. Galimi konfigūruojami TreeBagger klasifikatoriaus pagrindiniai parametrai – medžių skaičius, metodas (klasifikacijos ar regresijos), medžio genėjimo funkcija, atsitiktinai pasirenkamų bruožų skaičius kiekvienoje šakoje, konfiguruojamas šakų svoris ir t.t. Vienas iš gautų medžių, klasifikuojant bruožus išskirtus HOG metodu, pateiktas 2.7 pav.



2.7 pav. Vienas iš gautų sprendimų medžių

„MATLAB“ aplinkoje greitesnis (angl. *faster*) *R-CNN* detektorius apmokomas su funkcija *trainFasterRCNNObjectDetector*, tai vienas populiariausių gilaus mokymosi (angl. *deep learning*) algoritmų, objektų aptikimo srityje. Apmokymui reikalingi įėjimo duomenys:

- Treniravimo duomenys, pateikti kaip lentelė, pirmame stulpelyje nuorodos į failą, kituose stulpeliuose klasėms priskiriami dominantys nuotraukų regionai (ROI);
- Neuroninis tinklas – galima naudoti iš anksto ištreniruotą (angl. pretrained network) arba sukurti savo neuroninį tinklą iš norimo kiekio ir pobūdžio sluoksnių;
- Klasifikatoriaus parametrai – kiekvieno apmokymo etapo parametrai, kurių dėka galima nustatyti mokymosi greitį, konvoliucinio neuroninio tinklo epochų skaičių ir t.t.

2.5. Slenkantis langas ir nuotraukų piramidė

Norint aptikti teisėjo gestą, skirtingų išmatavimų nuotraukuose, nekeičiant slenkančio lango dydžio, reikia mažinti nuotraukas ir ieškoti kokie išmatavimai tinkami gesto aptikimui. Nuotraukų piramidė sukuriama su „MATLAB“ funkcija *imresize*. Originali nuotrauka mažinama 0,1 žingsniu iki ½ originalaus nuotraukos dydžio. Pavyzdžiui įkėlus nuotrauką, kurios išmatavimai 800x600 – sumažinama iki 720x540, tada 640x480, ..., galiausiai iki 400x300. Slenkantis langas, kurio išmatavimai 100x75, pateikiamas ant skirtingo dydžio nuotraukų 2.8 pav.

Slenkančio lango funkcija buvo realizuota *for* cikle. Įvertinus įkeltos nuotraukos išmatavimus atitinkamai, iš nuotraukos pločio atimamas lango plotis ir iš nuotraukos aukščio atimamas lango aukštis. Tokiu būdu slinkdamas langas neperžengia nuotraukos ribų. Norint sumažinti, bruožų vektorių skaičių galima padidinti lango slinkimo žingsnio (angl. *stride*) pikselių skaičių. Darbe buvo nuspręsta naudoti 10 pikselių žingsnį horizontaliai ir 5 pikselių žingsnį vertikaliai. Taigi, jeigu turime 800x600 nuotrauką ir slenkantį langą 100x75, langų telpančių nuotraukoje skaičius N išreiškiamas formule:

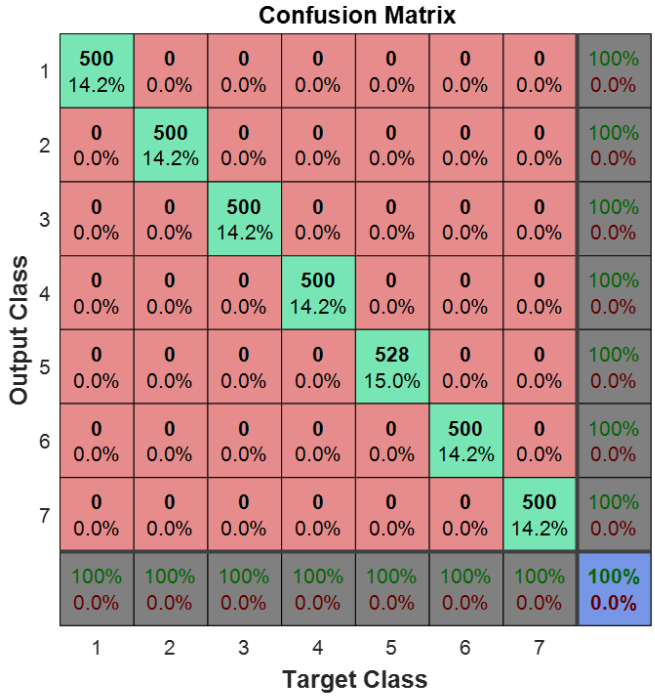
$$N = \left(\frac{Aukštis - Aukštis\_L}{Žingsnis\_A}\right) * \left(\frac{Plotis - Plotis\_L}{Žingsnis\_P}\right) = \left(\frac{700}{5}\right) * \left(\frac{525}{10}\right) = 140 * 52 = 7280. \quad (2.4)$$

2.8 pav. Slenkantis langas vaizduojamas ant skirtingų dydžių nuotraukų

3. Eksperimentiniai tyrimai

3.1. Klasifikatorių patikra

Buvo sukurti 6 klasifikatoriai su slenkančio lango funkcija ir skirtingais bruožų išskyrimo metodais, naudota RCNN objektų aptikimo funkcija „MATLAB aplinkoje. Norint įsitikinti, kad sukurti klasifikatoriai teisingai veikia, buvo nuspręsta patikrinti jų veikimą be slenkančio lango, naudojant tas pačias apmokymui naudotas nuotraukas. Iš viso buvo naudojamos 7 klasės: fonas ir 3 teisėjo rodomos komandos, teisėjui stovint nugara ir atsisukus priekiu į kamerą. Apmokymo nuotraukų skaičius – 3528, po 500 nuotraukų kiekvienai komandos klasei ir 528 nuotraukso fonui. Derinant ir ekperimentiškai testuojant klasifikatorių, buvo siekiama gauti optimalius klasifikavimo parametrus, kad visos nuotraukos būtų suklasifikuotos teisingai. Visi sukurti metodai – RF ir SVM, su bruožų išskyrimo metodais HOG, LBP ir PCA, suklasifikavo treniravimo duomenis 100 proc. teisingai. Vizualizacijai pateikta panašumo matrica (angl. *confusion matrix*), kur y ašyje matoma gauta klasė, o x ašyje iš tiesų esanti klasė, matrica vaizduojama 3.1 pav.

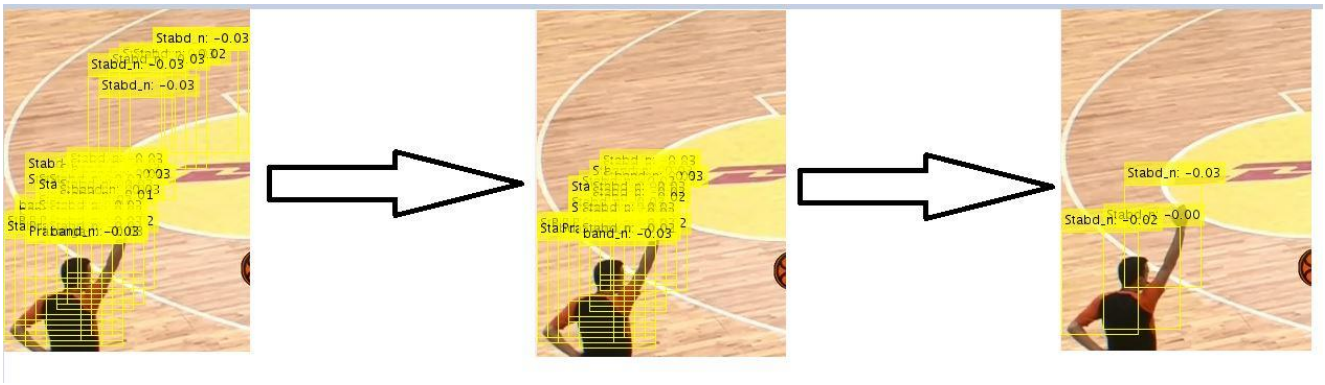


3.1 pav. Klasifikatorių panašumo matrica su testavimo duomenimis

3.2.Gestų aptikimo sistemos vizualizacija

Krepšinio teisėjo gestų sistema buvo realizuota 7 metodais: *HOG+SVM, HOG+RF, PCA+SVM, PCA+RF, LBP+SVM, LBP+RF ir FasterRCNN*. Gestų atpažinimo sistemoje, svarbu ne tik koks gestas buvo aptpažintas, bet ir kokia jo pozicija nuotraukoje – ar atpažintas gestas nebuvo tik atsitiktinai komandai priskirta fono dalis. Šiam tikslui pasiekti, realizuotas slenkantis langas vaizduojamas ant nuotraukos.

Praktikoje, kai naudojamas slenkantis langas, klaidingai atpažintos komandos iškerpamos ir priskiriamos fono klasei, tada klasifikatorius apmokomas iš naujo ir klaidingai atpažintų objektų (angl. *false positives*) skaičius yra sumažinamas. Šis metodas taikomas kiekvienai testuojamai nuotraukai. Po šių dviejų etapų, klaidingai atpažįstamų komandų skaičius ženkliai sumažėja (3.2 pav pirma ir antra nuotrauka iš kairės), tačiau daug langų persidengia ir sunku įžvelgti atpažinto gesto poziciją. Persidengiančių langų eliminacijai naudojamas netiesioginio slopinimo (angl. *non maximal suppression*) metodas, „MATLAB“ aplinkoje su funkcija *selectStrongestBbox*. Ir gaunamas galutinis gestų atpažinimo sistemos išvedamas atsakymas (3.2 pav. dešinė nuotrauka). Langai sujungiami, jeigu jie priklauso tai pačiai klasei ir persidengia bent 30 proc.



3.2 pav. Teisėjo gestų atpažinimo etapai

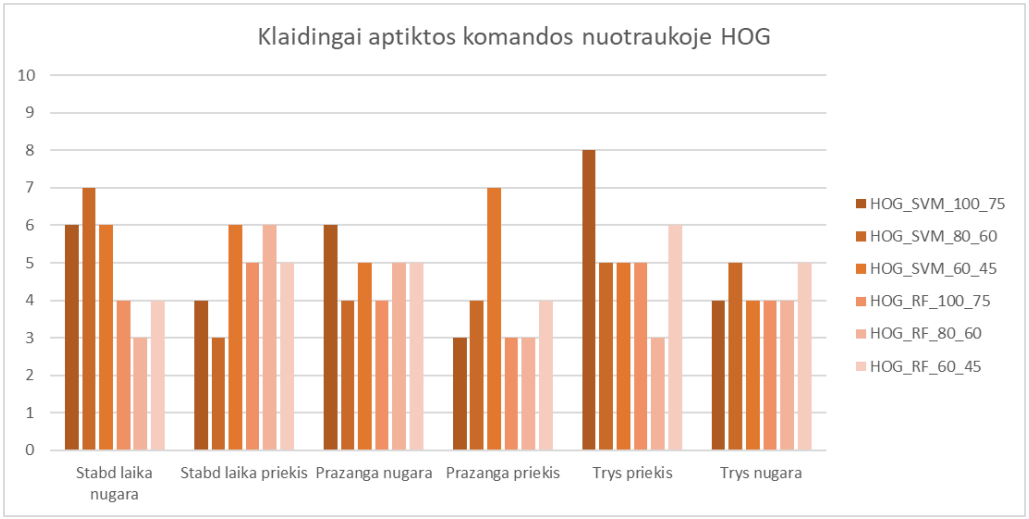
CNN deketorius buvo realizuotas naudojant „MATLAB“ funkciją *trainRCNNObjectDetector*. Su funkcija detector ant nuotraukos išvedamas gestas su patikimumo (angl. *confidence*) taškais. Sukurtoje SVM ir RF gestų aptikimo sistemoje išvedami visi rezultatai virš taškų ribos. 3.3 pav. vaizduojami RCNN algoritmu aptikti teisingi ir klaidingi gestai.

3.3 pav. Teisingai ir klaidingai suklasifikuoti gestai RCNN algoritmu

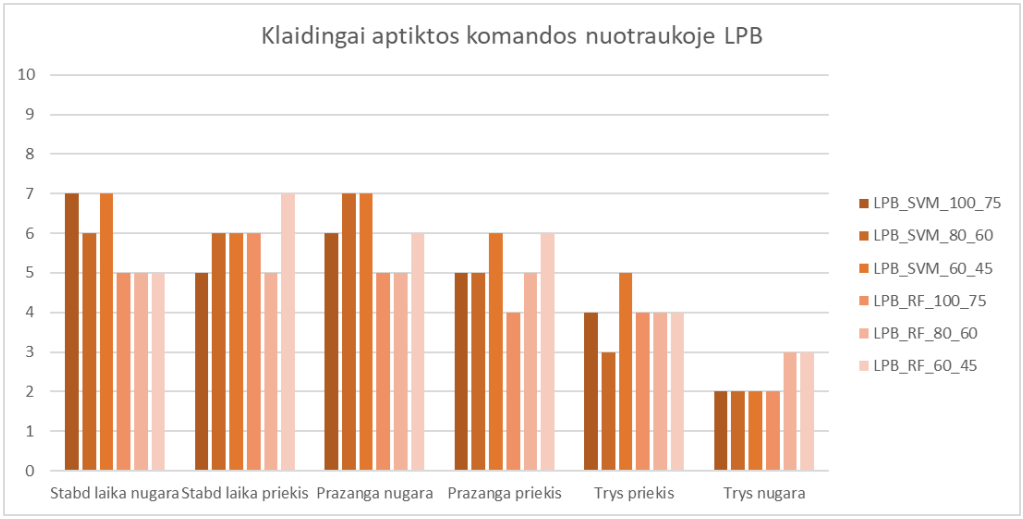
3.3. Krepšinio teisėjo gestų atpažinimo sistemos tyrimas

Norint išsiaiškinti sistemos patikimumą ir tikslumą buvo atliktas jos tyrimas. Sistema apmokyta atpažinti 6 klases (3 gestus): trys taškai – bandymas, stabdyti laiką ir stabdyti laiką – pražanga, klasifikuojama skirtingai, kai teisėjas stovi atsisukęs į kamerą ir kai nusisukęs. Testavimui iš atsitiktinių varžybų buvo iškirpta 60 kadrų, kuriuose rodomi minėti gestai, su skirtingomis teisėjo pozicijomis kameros atžvilgiu. Taigi vienai klasei testuoti naudojama 10 nuotraukų. Sistema testuojama keičiant slenkančio lango dydį, naudojami 3 dydžiai: 100x75, 80x60, 60x45.

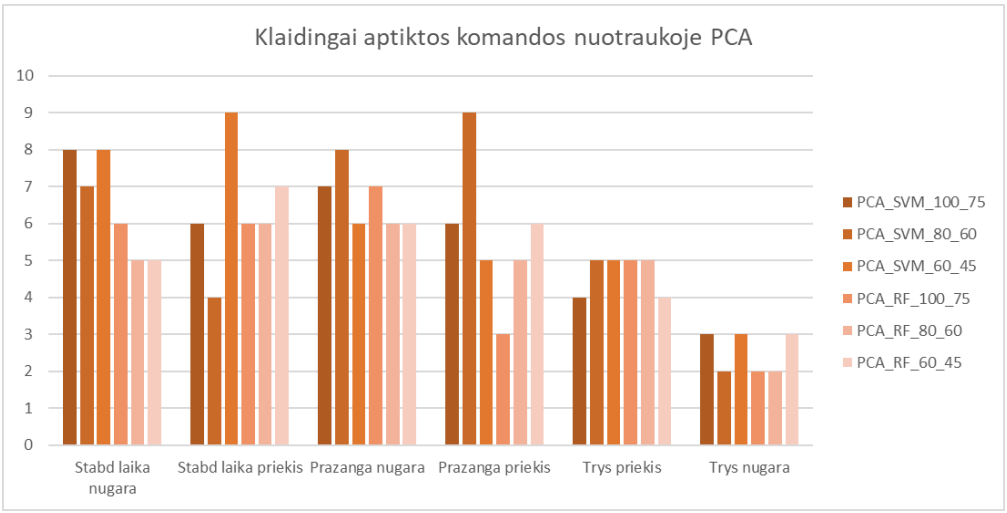
Testuojant RF ir SVM klasifikatorius, svarbus klasifikatoriaus vertinimo kriterijus yra neteisingai atpažintas gestas nuotraukoje (angl. *False positive per image*). Neteisingai atpažintu gestu laikomas klasifikatoriaus langas, kuriame nėra teisėjo ir rodomo gesto (tai gali būti parketas, žiūrovas, žaidėjas ar kažkuris kitas fono elementas). Kuo mažesnis šio kriterijaus rodiklis, tuo klasifikatoriaus veikimas laikomas geresniu. Toliau pateikiami klaidingai atpažintų komandų grafikai, su skirtingais langų dydžiais ir klasifikatorių/bruožų išskyrimo kombinacijomis. X ašyse yra nurodytos bandomos atpažinti komandos, o y ašyse pateiktas, klaidingai atpažintų komandų vidurkis, iš testavimo duomenų imties. Klaidingai aptiktų komandų grafikai pateikti 3.4 , 3.5, 3.6 pav.



3.4 pav. Klaidingai aptiktos komandos HOG bruožų išskyrimo metodu

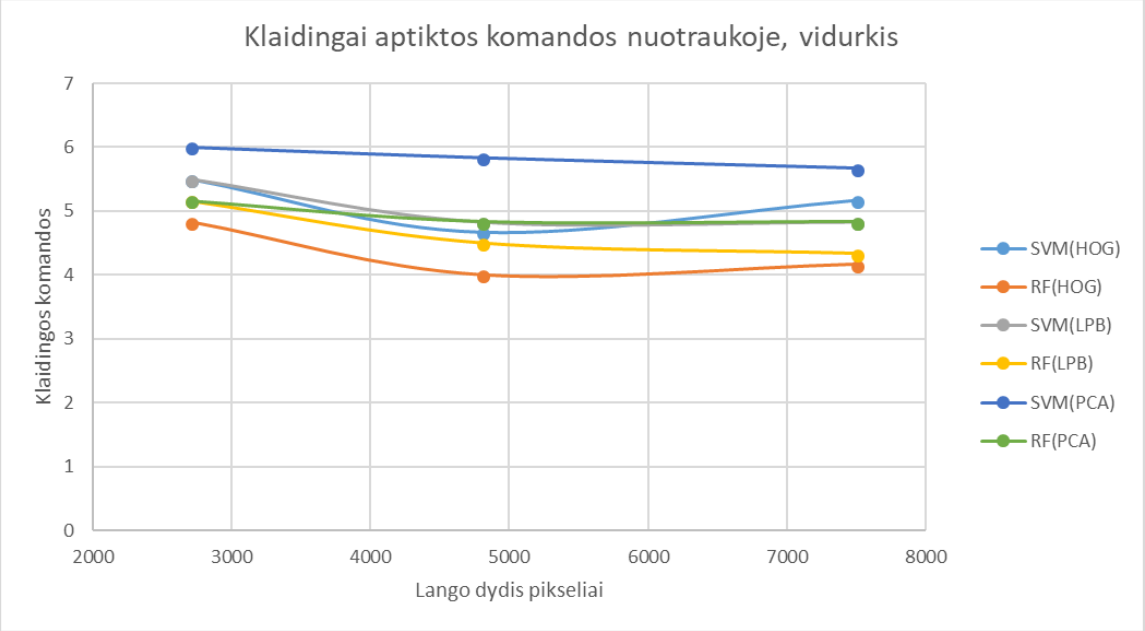


3.5 pav. Klaidingai aptiktos komandos LBP bruožų išskyrimo metodu



3.6 pav. Klaidingai aptiktos komandos PCA bruožų išskyrimo metodu

Iš grafikų matome, kad klaidingai aptiktų komandų kiekis nuotraukose svyruoja nuo 2 iki 9. Taip pat, daugiausiai klaidingai atpažįstamos gestų nuotraukos, kur rodomas stabdyti laiką gestas, o mažiausiai – trys taškai bandymas. Atsitiktinių miškų klasifikatorius, mažiau komandų atpažino klaidingai. Lango dydžio pokyčiai, taip pat turėjo įtakos klaidingai aptiktų gestų skaičiui. Todėl buvo suvidurkintos visos klaidingai atpažintos komandos ir jų priklausomybė nuo lango dydžio vaizduojama 3.7 pav. X ašyje pateikiami langų dydžiai pikseliais – 7500, 4800 ir 2700.



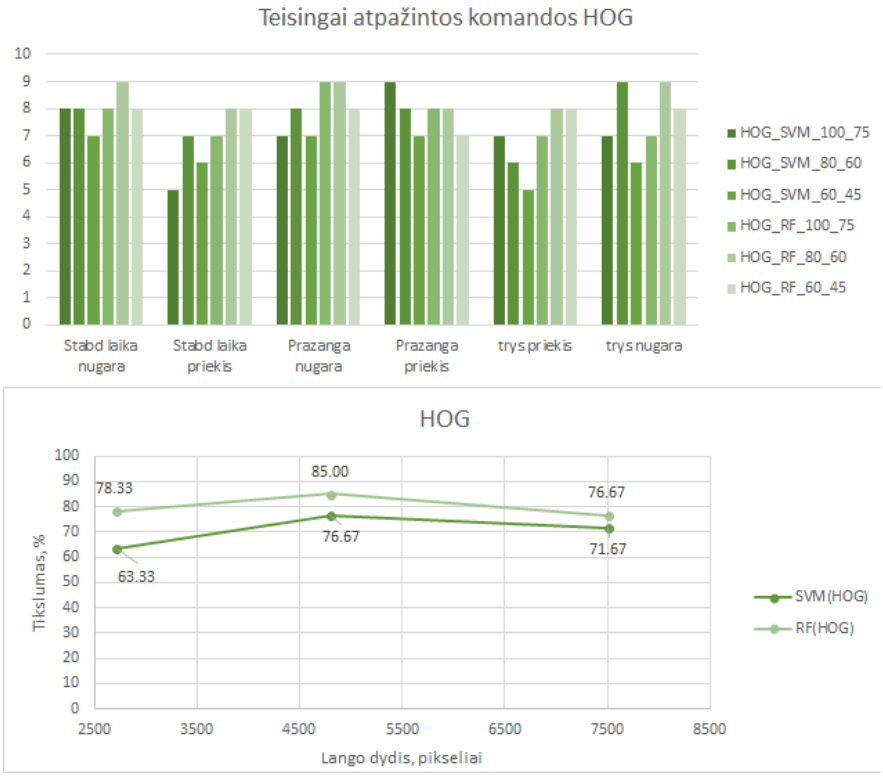
3.7 pav. Klaidingai aptiktų komandų nuotraukoje vidurkis

Vidutiniškai nuotraukoje aptinkami 6-4 klaidingi gestai. Vidutiniškai daugiausiai klaidingų gestų užregistruojama su SVM+PCA klasifikatoriumi, o mažiausiai RF+HOG. Mažinant aptikimo lango dydį, sumažėja ir bruožų vektoriaus ilgis – klasifikuojami duomenys atsparesni triukšmui. Todėl mažinant lango dydį su visomis klasifikatorių/bruožų išskyrimo metodų kombinacijomis, aptinkamų klaidingų gestų skaičius mažėja. Geriausi rezultatai gauti, kai lango dydis buvo 80x60.

Dar vienas svarbus sistemos vertinimo kriterijus tikslumas. Gestas laikomas atpažintu teisingai, kai priskirta klasė ir lango pozicija išvedama teisingai. Sistemos testavimui, kaip ir minėta buvo naudojama 60 nuotraukų. Jeigu laikysime, kad N teisingai suklasifikuotos komandos:

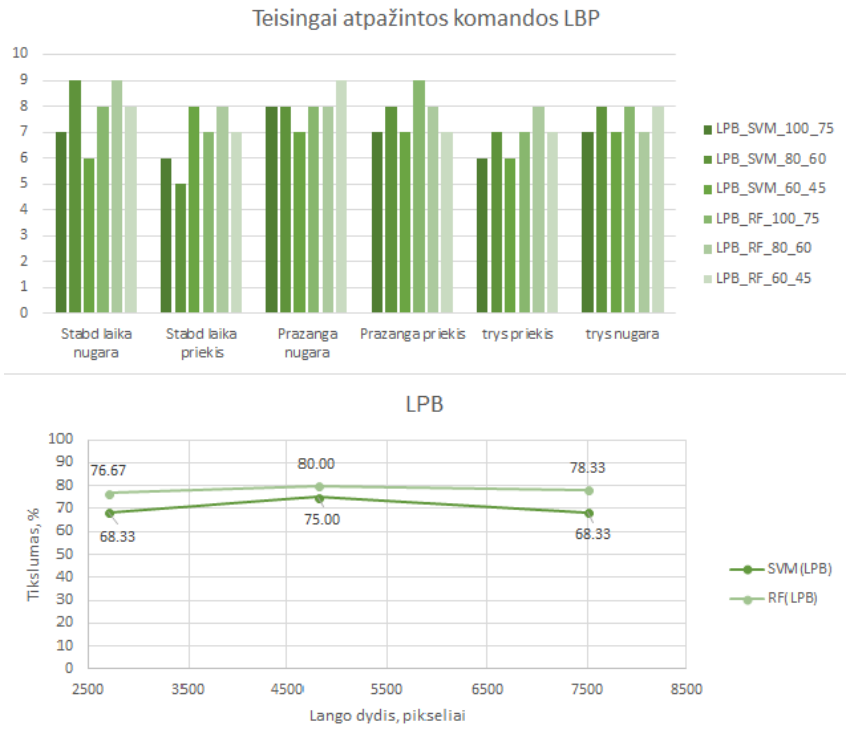
$$Tikslumas = \frac{N * 100}{60} \tag{3.1}$$

Teisingai atpažintų komandų skaičius (maksimalus – 10) ir klasifikavimo tikslumas, su skirtingais klasifikatoriaus ir lango dydžiais, HOG bruožų išskyrimo metodu pateikiamas 3.8 pav.



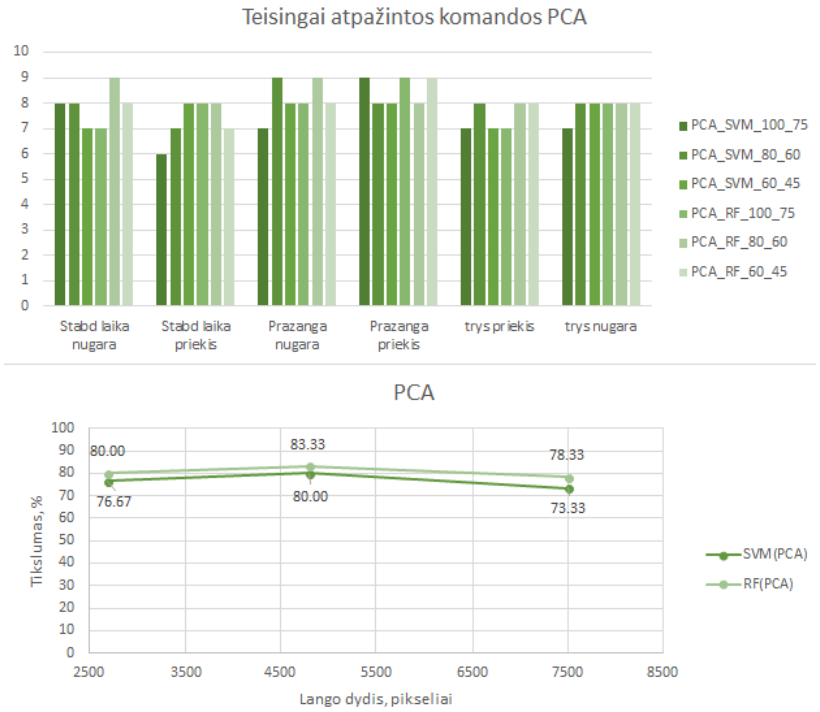
3.8 pav. Atpažinimo tikslumas naudojant HOG

Atpažintų komandų skaičius HOG bruožų išskyrimo metodu svyruoja nuo 5 iki 9. Tiksliausiai atpažįstama pražangos komanda, daugiausiai klystama su trys taškai – bandymas komanda. Geriausias abiejų klasifikatorių tikslumas, kai lango dydis 4800 pikselių. Geriausias gautas tikslumas su HOG+RF, kai lango dydis 80x60 pikselių, blogiausias – HOG+SVM, kai lango dydis 100x75. Toliau pateikiamas klasifikavimo tikslumas naudojant LBP bruožų išskyrimo metodą 3.9 pav.



3.9 pav. Atpažinimo tikslumas naudojant LBP

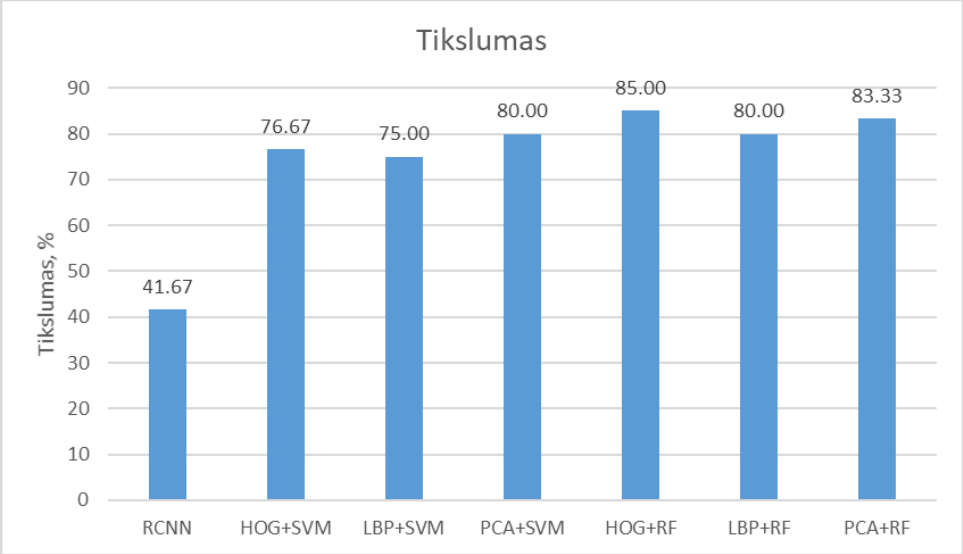
Atpažintų komandų skaičius LBP bruožų išskyrimo metodu svyruoja nuo 5 iki 9. Konkrečių komandų atpažinimų skaičius panašiai pasiskirstęs, kaip ir HOG metodu. Gautas geriausias tikslumas yra prastesnis 5 procentais lyginant su HOG, tačiau išlieka tendencija, kad RF klasifikatorius yra tikslesnis ir tiksliausiai gestai atpažįstami su 4800 pikselių langu. Toliau tikrinamas tikslumas su PCA 3.10 pav.



3.10 pav. Atpažinimo tikslumas naudojant PCA

Naudojant, PCA atpažintų koamdnų skaičius pasisikirstė tolygiau lyginant su kitais dviem metodais ir svyruoja nuo 6 iki 9. Tačiau, nors ir nežymiai, bet geresni rezultatai buvo gauti su RF klasifikatoriumi ir geriausi atpažinimo rezultatai gauti, kai lango dydis 4800 pikselių.

Buvo išrinkti geriausi RF ir SVM tikslumo rezultatai ir lyginami su RCNN klasifikatoriumi, rezultatai pateikti 3.11 pav.



3.11 pav. Visų klasifikatorių tikslumo palyginimas

Taip pat buvo vertinamas vidutinis klasifikatorių apmokymo ir klasifikavimo laikas, apibendrinti tyrimo rezultatai pateikia 3.1 lentelėje.

3.1 lentelė Klasifikatorių parametrų palyginimas

Klasifikatorius	Tikslumas, %	Apmokymo trukmė	Klasifikavimo trukmė
HOG+SVM	76,67	2 min 25s	37s
LBP+SVM	75	3min 10s	45s
PCA+SVM	80	3 min	55 s
HOG+RF	85	2min 1s	39s
LBP+RF	80	2min 40s	30s
PCA+RF	83,33	1min 50s	25 s
R-CNN	41,67	3d 8h 13min 10s	2s

Iš lentelėje pateiktų duomenų matome, kad tiksliausiai gestus klasifikuoja RF+HOG klasifikatorius, o prasčiausiai R-CNN klasifikatorius, taip yra dėl per mažo treniravimo duomenų kiekio. Dėl naudojamos senos techninės įrangos R-CNN apmokymas truko ilgiau nei 3 dienas. Tačiau greičiausiai klasifikuoja duomenis būtent R-CNN klasifikatorius. Visų kitų klasifikatorių apmokymo ir klasifikavimo trukmės yra labai panašios.

IŠVADOS

1. Tiksliausiai – 85% krepšinio teisėjo gestų, atpažinti naudojant orientuotų gradientų histogramos bruožus ir atsitiktinių miškų klasifikatorių.
2. Gautas visų klasifikatorių tikslumas panašus – svyruoja nuo 75 iki 85%, išskyrus CNN klasifikatorių, kurio tikslumas siekė tik 42%. Toks rezultatas gautas, dėl nepakankamo treniravimo duomenų kiekio.
3. Mažiausias klaidingai atpažintų komandų skaičius nuotraukoje, naudojant atsitiktinių miškų klasifikatorių ir HOG bruožų išskyrimo metodą – apytiksliai 4 klaidos per nuotrauką. Padidinus treniravimo duomenų kiekį, klaidų kiekis galėtų būti ženkliai sumažintas.
4. Mažinant slenkančio lango dydį, tiksliausi rezultatai gaunami naudojant 80x60 dydžio langą, su visomis klasifikatorių ir bruožų išskyrimo metodų kombinacijomis.
5. Įvertinus klasifikatorių apmokymo ir gestų klasifikavimo laiką, RF ir SVM klasifikatorių apmokymo trukmė svyruoja nuo 2min iki 3min, o RCNN klasifikatoriaus apmokymo laikas siekė 3d 8h. RCNN klasifikavo gestus greičiausiai - 2s vienai nuotraukai. SVM ir RF klasifikavimo laikas svyravo nuo 25s iki 55s.

INFORMACIJOS ŠALTINIŲ SĄRAŠAS

1. RAFAEL C. GONZALEZ, RICHARD E. WOODS Digital Image Processing Third Edition, 2008. pp. 1-8. [Žiūrėta: 2017-04-15]. Prieiga per internetą: http://web.ipac.caltech.edu/staff/fmasci/home/astro_refs/Digital_Image_Processing_3rdEd_truncated.pdf
2. ROGER L. EASTON, JR., Fundamentals of Digital Image Processing, 2010. pp. 7-20 [Žiūrėta: 2017-04-15]. Prieiga per internetą: https://www.cis.rit.edu/class/simg361/Notes_11222010.pdf
3. J.R. PARKER, Algorithms for Image Processing and Computer Vision 2011. pp. 1 – 15 [Žiūrėta: 2017-04-20]. Prieiga per internetą: http://www.manalhelal.com/Books/crol/Algorithms%20for%20Image%20Processing%20and%20Computer%20Vision_2011.pdf
4. GAURAV KUMAR, Pradeep Kumar Bhatia, A Detailed Review of Feature Extraction in Image Processing Systems, IEEE, 2014. [Žiūrėta: 2017-04-20]. Prieiga per internetą: https://ieeexplore.ieee.org/document/6783417/
5. MANEELA JAIN, PUSHPENDRA SINGH TOMAR, Review of Image Classification Methods and Techniques, IEEE, 2013, ISSN: 2278-0181. [Žiūrėta: 2017-04-25]. Prieiga per internetą: https://www.ijert.org/.../4730/review-of-image-classification-methods-and-techniques
6. MARK S. NIXON, ALBERTO S. AGUADO, Feature Extraction and Image Processing Second Edition, 2008 pp. 132 – 200. [Žiūrėta: 2017-04-25]. Prieiga per internetą: https://www.ppgia.pucpr.br/~facon/ComputerVisionBooks/2002FeatureExtractionAndImageProcessing.pdf
7. LIETUVOS KREPŠINIO FEDERACIJOS TEISĖJŲ ASOCIACIJA, Oficialios Krepšinio taisyklės, 2014, Patvirtinta FIBA [Žiūrėta: 2017-04-25]. Prieiga per internetą: http://www.krepsinioteisejas.lt/index.php/pradzia/visos-naujienos/198-fiba-krepsinio-taisykliu-interpretaciju-redakcija
8. CHAITALI DHAWARE , MRS. K. H. WANJALE, Survey On Image Classification Methods In Image Processing, 2016, ISSN: 2347-8578. [Žiūrėta: 2017-04-25]. Prieiga per internetą: http://www.ijcstjournal.org/volume-4/issue-3/IJCST-V4I3P40.pdf
9. SAMER HIJAZI, RISHI KUMAR, CHRIS ROWEN, Using Convolutional Neural Networks for Image Recognition, 2015, pp. 1 – 2. [Žiūrėta: 2017-04-25]. Prieiga per internetą: https://pdfs.semanticscholar.org/bbf7/b5bdc39f9b8849c639c11f4726e36915a0da.pdf

10. P.JEEVITHA, Dr. P. GANESH KUMAR, Spatial Information Based Image Classification Using Support Vector Machine, 2014, ISSN:2320-9801. [Žiūrėta: 2017-04-25]. Prieiga per internetą: http://www.ijircce.com/upload/2014/nsci14/3_DM02.pdf
11. M. MADHUBALA, G. RAVINDRA BABU, DR. S.K. MOHAN RAO, KNOWLEDGE BASED IMAGE CLASSIFICATION USING DECISION TREE ALGORITHM, 2007. [Žiūrėta: 2017-04-25]. Prieiga per internetą: https://pdfs.semanticscholar.org/72bc/0a69fcbddd46fffcc24d82c59c844cd11d27.pdf
12. Donatas Dervinis „ Vaizdų apdorojimas“ Mokomoji knyga, 2012 pp. 17-24 [Žiūrėta: 2018-05-05]. Prieiga per internetą: https://www.ebooks.ktu.lt/eb/451/vaizdu-apdorojimas/
13. „Histograms of Oriented Gradients“, Carlo Tomasi, 2017 [Žiūrėta: 2018-05-07]. Prieiga per internetą: https://pdfs.semanticscholar.org/8086/99bacf0cc91a38fa77b6af5a9f91dc25d127.pdf
14. M. Pietikäinen et al., Computer Vision Using Local Binary Patterns, 2011 [Žiūrėta: 2018-05-10]. Prieiga per internetą: https://www.springer.com/cda/content/document/cda_downloaddocument/9780857297471-c2.pdf?SGWID=0-0-45-1153741-p174122174
15. Local Binary Patterns [Žiūrėta: 2018-05-10]. Prieiga per internetą: http://www.scholarpedia.org/article/Local_Binary_Patterns
16. „Principal Component analysis in image processing“, M. Mudrov´a, A. Proch´azka [Žiūrėta: 2018-05-10]. Prieiga per internetą: https://pdfs.semanticscholar.org/76a7/fc9d87736c8383576865cf50403e53e74848.pdf
17. Feature extraction using PCA, [Žiūrėta: 2018-05-10]. Prieiga per internetą: http://www.visiondummy.com/2014/05/feature-extraction-using-pca/
18. „Multi-Class Support Vector Machine“, Zhe Wang and Xiangyang Xue, 2014 [Žiūrėta: 2018-05-11]. Prieiga per internetą: https://www.springer.com/cda/content/document/cda_downloaddocument/9783319022994-c1.pdf?SGWID=0-0-45-1446422-p175468473
19. PUBLIC: A Decision Tree Classifier that Integrates Building and Pruning, 2000, [Žiūrėta: 2018-05-11]. Prieiga per internetą: http://sci2s.ugr.es/keel/pdf/algorithm/articulo/PUBLIC.pdf
20. „Split Selection Methods for Classification Trees.” Loh, W.Y. ir Y.S. Shih, 1997, [Žiūrėta: 2018-05-11]. Prieiga per internetą: http://www3.stat.sinica.edu.tw/statistica/oldpdf/A7n41.pdf

21. „Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks“, Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, 2016, [Žiūrėta: 2018-05-11]. Prieiga per internetą: https://arxiv.org/pdf/1506.01497.pdf
22. Eigenfaces face recognition (MATLAB), [Žiūrėta: 2018-05-11]. Prieiga per internetą: https://blog.cordiner.net/2010/12/02/eigenfaces-face-recognition-matlab/
23. „MATLAB“ dokumentacija, [Žiūrėta: 2018-05-15]. Prieiga per internetą: https://se.mathworks.com/help/

PRIEDAI

Priedas nr. 1, Matlab .m programinis kodas, slenkančio lango klasifikatorius

```
 %% Clear data
    clear all;
    close all;
    clc;

%% Uzkrauti training data ir ivertinti vieno paveikslelio dydi
Gestai   = fullfile('F:\TesejoGestaiOriginal_Sorted_NAUDOTI_ISKIRPTI _HOG_100_75');

Treniravimo_duomenys = imageDatastore(Gestai,  'IncludeSubfolders', true, 'LabelSource',
'foldernames');

img = readimage(Treniravimo_duomenys,550);
%img2 = readimage(Treniravimo_duomenys,1000);
%preprocessing
img = rgb2gray(img);
img = histeq(img);

[h,l,c] = size(img);


%% Isskirti HOG bruozus
cellSize = [4 4];

numImages = numel (Treniravimo_duomenys.Files); %susumuojami visu treniravimo duomenu nuotr

tic
%isskiriu is visu duomenu HOG bruozus
for i = 1:numImages
    img = readimage(Treniravimo_duomenys, i);

    img = rgb2gray(img);
    img=histeq(img);


    treniravimo_bruozai(i, :) = extractHOGFeatures(img, 'CellSize', cellSize);
end

%lapeliai kiekvienai klaisei
treniravimo_lapeliai = Treniravimo_duomenys.Labels;

%% SVM klasifikatorius
classifier = fitcecoc(treniravimo_bruozai, treniravimo_lapeliai);
toc

%% testuojam klasifikatoriu slenkanciu langu
img_test = imread('F:\TEST_DATA_NAUDOTI\Stabdytilaika_back\test2.jpg');

%paveikselio sumazinimas piramide
img_test2 = imresize(img_test, 0.9);
img_test3 = imresize(img_test, 0.8);
img_test4 = imresize(img_test, 0.7);
img_test5 = imresize(img_test, 0.6);
img_test6 = imresize(img_test, 0.5);

img_test_gray = rgb2gray(img_test);
img_test_gray2 = rgb2gray(img_test2);
img_test_gray3 = rgb2gray(img_test3);
img_test_gray4 = rgb2gray(img_test4);
img_test_gray5 = rgb2gray(img_test5);
img_test_gray6 = rgb2gray(img_test6);

%preprocessing steps
img_test_gray = histeq(img_test_gray);
img_test_gray2 = histeq(img_test_gray2);
img_test_gray3 = histeq(img_test_gray3);
img_test_gray4 = histeq(img_test_gray4);
img_test_gray5 = histeq(img_test_gray5);
img_test_gray6 = histeq(img_test_gray6);

% nustatome test paveikslelio dydzius visai piramidei
[tl, th, tc] = size(img_test_gray);
[tl2, th2, tc2] = size(img_test_gray2);
```

```
[tl3, th3, tc3] = size(img_test_gray3);
[tl4, th4, tc4] = size(img_test_gray4);
[tl5, th5, tc5] = size(img_test_gray5);
[tl6, th6, tc6] = size(img_test_gray6);

wh =h-1;
wl=l-1;

dydis = 1;%bruozu masyvo dydis
dydis2 = 1;
dydis3 = 1;
dydis4 = 1;
dydis5 = 1;
dydis6 = 1;


%slenkantis langas
for row=1:5:tl-wh
    for col=1:10:th-wl
        blokas = img_test_gray(row:row+wh,col:col+wl,:);
        posx(dydis, :)=col;
        posy(dydis, :)=row;
        treniravimo_bruozai_test(dydis, :) = extractHOGFeatures(blokas, 'CellSize', cellSize);

        dydis = dydis + 1;

    end
end

for row=1:5:tl2-wh
    for col=1:10:th2-wl
        blokas = img_test_gray2(row:row+wh,col:col+wl,:);
        posx2(dydis2, :)=col;
        posy2(dydis2, :)=row;
        treniravimo_bruozai_test2(dydis2, :) = extractHOGFeatures(blokas, 'CellSize', cellSize);

        dydis2 = dydis2 + 1;

    end
end

for row=1:5:tl3-wh
    for col=1:10:th3-wl
        blokas = img_test_gray3(row:row+wh,col:col+wl,:);
        posx3(dydis3, :)=col;
        posy3(dydis3, :)=row;
        treniravimo_bruozai_test3(dydis3, :) = extractHOGFeatures(blokas, 'CellSize', cellSize);

        dydis3 = dydis3 + 1;

    end
end

for row=1:5:tl4-wh
    for col=1:10:th4-wl
        blokas = img_test_gray4(row:row+wh,col:col+wl,:);
        posx4(dydis4, :)=col;
        posy4(dydis4, :)=row;
        treniravimo_bruozai_test4(dydis4, :) = extractHOGFeatures(blokas, 'CellSize', cellSize);

        dydis4 = dydis4 + 1;

    end
end

for row=1:5:tl5-wh
    for col=1:10:th5-wl
        blokas = img_test_gray5(row:row+wh,col:col+wl,:);
        posx5(dydis5, :)=col;
        posy5(dydis5, :)=row;
        treniravimo_bruozai_test5(dydis5, :) = extractHOGFeatures(blokas, 'CellSize', cellSize);

        dydis5 = dydis5 + 1;

    end
end

for row=1:5:tl6-wh
```

```
    for col=1:10:th6-wl
        blokas = img_test_gray6(row:row+wh,col:col+wl,:);
        posx6(dydis6, :)=col;
        posy6(dydis6, :)=row;
        treniravimo_bruozai_test6(dydis6, :) = extractHOGFeatures(blokas, 'CellSize', cellSize);

        dydis6 = dydis6 + 1;

    end
end

%klasifikuojam duomenis visiem piramides duomenims
[predictedLabels, score] = predict(classifier, treniravimo_bruozai_test);
[predictedLabels2, score2] = predict(classifier, treniravimo_bruozai_test2);
[predictedLabels3, score3] = predict(classifier, treniravimo_bruozai_test3);
[predictedLabels4, score4] = predict(classifier, treniravimo_bruozai_test4);
[predictedLabels5, score5] = predict(classifier, treniravimo_bruozai_test5);
[predictedLabels6, score6] = predict(classifier, treniravimo_bruozai_test6);

% surasti kiek kokiu yra labels ir koks ju numeris koordinatems uzdeti
dydis_label = 1;
dydis_label2 = 1;
dydis_label3 = 1;
dydis_label4 = 1;
dydis_label5 = 1;
dydis_label6 = 1;


%numatyti lapeliai visai piramidei
num_predict = numel(predictedLabels);
num_predict2 = numel(predictedLabels2);
num_predict3 = numel(predictedLabels3);
num_predict4 = numel(predictedLabels4);
num_predict5 = numel(predictedLabels5);
num_predict6 = numel(predictedLabels6);

slenkstis = single(-0.03);


%nustatomos piramides lapeliu koordinates
%orginalus dydis
for k = 1:num_predict
    prediction(k,:) = string(predictedLabels(k,1));
    if strcmp(prediction(k,1),'trystaskaibandymas_front') == 1
        if score(k,7) > slenkstis
        pozicija_lapelio(dydis_label,1) = posx(k,1);
        pozicija_lapelio(dydis_label,2) = posy(k,1);
        pozicija_lapelio(dydis_label,3) = wl;
        pozicija_lapelio(dydis_label,4) = wh;
        score_NMS1(dydis_label,1) = score(k,7); %taskai reikalingi NMS
        label_str{dydis_label} = ['band_p: ' num2str(score(k,7),'%0.2f')];  %5ame stulpeli yra du
taskai klases NEGLOSS
        dydis_label = dydis_label +1;
        end
     elseif strcmp(prediction(k,1),'Stabdlaikaprazanga_back' ) == 1
        if score(k,1) > slenkstis
        pozicija_lapelio(dydis_label,1) = posx(k,1);
        pozicija_lapelio(dydis_label,2) = posy(k,1);
        pozicija_lapelio(dydis_label,3) = wl;
        pozicija_lapelio(dydis_label,4) = wh;
        score_NMS1(dydis_label,1) = score(k,1); %taskai reikalingi NMS
        label_str{dydis_label} = ['Prazanga_n: ' num2str(score(k,1),'%0.2f')]; %6ame stulpeli yra
trys taskai bandymas klases NEGLOSS
        dydis_label = dydis_label +1;
        end
     elseif strcmp(prediction(k,1), 'Stabdlaikaprazanga_front' ) == 1
        if score(k,2) > slenkstis
        pozicija_lapelio(dydis_label,1) = posx(k,1);
        pozicija_lapelio(dydis_label,2) = posy(k,1);
        pozicija_lapelio(dydis_label,3) = wl;
        pozicija_lapelio(dydis_label,4) = wh;
        score_NMS1(dydis_label,1) = score(k,2); %taskai reikalingi NMS
        label_str{dydis_label} = ['Prazanga_p: ' num2str(score(k,2),'%0.2f')]; %6ame stulpeli yra
trys taskai bandymas klases NEGLOSS
        dydis_label = dydis_label +1;
        end
     elseif strcmp(prediction(k,1), 'Stabdytilaika_back' ) == 1
        if score(k,3) > slenkstis
```

```
        pozicija_lapelio(dydis_label,1) = posx(k,1);
        pozicija_lapelio(dydis_label,2) = posy(k,1);
        pozicija_lapelio(dydis_label,3) = wl;
        pozicija_lapelio(dydis_label,4) = wh;
        score_NMS1(dydis_label,1) = score(k,3); %taskai reikalingi NMS
        label_str{dydis_label} = ['Stabd_n: ' num2str(score(k,3),'%0.2f')]; %4ame stulpeli yra
stabdyti laika klases NEGLOSS
        dydis_label = dydis_label +1;
        end
      elseif strcmp(prediction(k,1), 'Stabdytilaika_front' ) == 1
        if score(k,4) > slenkstis
        pozicija_lapelio(dydis_label,1) = posx(k,1);
        pozicija_lapelio(dydis_label,2) = posy(k,1);
        pozicija_lapelio(dydis_label,3) = wl;
        pozicija_lapelio(dydis_label,4) = wh;
        score_NMS1(dydis_label,1) = score(k,4); %taskai reikalingi NMS
        label_str{dydis_label} = ['Stabd_p: ' num2str(score(k,4),'%0.2f')]; %4ame stulpeli yra
stabdyti laika klases NEGLOSS
        dydis_label = dydis_label +1;
        end
      elseif strcmp(prediction(k,1), 'trystaskaibandymas_back' ) == 1
        if score(k,6) > slenkstis
        pozicija_lapelio(dydis_label,1) = posx(k,1);
        pozicija_lapelio(dydis_label,2) = posy(k,1);
        pozicija_lapelio(dydis_label,3) = wl;
        pozicija_lapelio(dydis_label,4) = wh;
        score_NMS1(dydis_label,1) = score(k,6); %taskai reikalingi NMS
        label_str{dydis_label} = ['band_n: ' num2str(score(k,6),'%0.2f')]; %3ame stulpeli yra
stabdyti laika prazanga klases NEGLOSS
        dydis_label = dydis_label +1;
        end
    else

    end


end
%dydis sumazintas 0.1 karto
for k = 1:num_predict2
    prediction(k,:) = string(predictedLabels2(k,1));
    if strcmp(prediction(k),'trystaskaibandymas_front') == 1
        if score2(k,7) > slenkstis
        pozicija_lapelio2(dydis_label2,1) = posx(k,1);
        pozicija_lapelio2(dydis_label2,2) = posy(k,1);
        pozicija_lapelio2(dydis_label2,3) = wl;
        pozicija_lapelio2(dydis_label2,4) = wh;
        score_NMS2(dydis_label2,1) = score2(k,7); %taskai reikalingi NMS
        label_str2{dydis_label2} = ['band_p: ' num2str(score2(k,7),'%0.2f')];  %5ame stulpeli yra
du taskai klases NEGLOSS
        dydis_label2 = dydis_label2 +1;
        end
     elseif strcmp(prediction(k),'Stabdlaikaprazanga_back' ) == 1
        if score2(k,1) > slenkstis
        pozicija_lapelio2(dydis_label2,1) = posx(k,1);
        pozicija_lapelio2(dydis_label2,2) = posy(k,1);
        pozicija_lapelio2(dydis_label2,3) = wl;
        pozicija_lapelio2(dydis_label2,4) = wh;
        score_NMS2(dydis_label2,1) = score2(k,1); %taskai reikalingi NMS
        label_str2{dydis_label2} = ['Prazanga_n: ' num2str(score2(k,1),'%0.2f')]; %6ame stulpeli
yra trys taskai bandymas klases NEGLOSS
        dydis_label2 = dydis_label2 +1;
        end
     elseif strcmp(prediction(k), 'Stabdlaikaprazanga_front' ) == 1
        if score2(k,2) > slenkstis
        pozicija_lapelio2(dydis_label2,1) = posx(k,1);
        pozicija_lapelio2(dydis_label2,2) = posy(k,1);
        pozicija_lapelio2(dydis_label2,3) = wl;
        pozicija_lapelio2(dydis_label2,4) = wh;
        score_NMS2(dydis_label2,1) = score2(k,2); %taskai reikalingi NMS
        label_str2{dydis_label2} = ['Prazanga_p: ' num2str(score2(k,2),'%0.2f')]; %6ame stulpeli
yra trys taskai bandymas klases NEGLOSS
        dydis_label2 = dydis_label2 +1;
        end
     elseif strcmp(prediction(k), 'Stabdytilaika_back' ) == 1
        if score2(k,3) > slenkstis
        pozicija_lapelio2(dydis_label2,1) = posx(k,1);
        pozicija_lapelio2(dydis_label2,2) = posy(k,1);
        pozicija_lapelio2(dydis_label2,3) = wl;
```

```
        pozicija_lapelio2(dydis_label2,4) = wh;
        score_NMS2(dydis_label2,1) = score2(k,3); %taskai reikalingi NMS
        label_str2{dydis_label2} = ['Stabd_n: ' num2str(score2(k,3),'%0.2f')]; %4ame stulpeli yra
stabdyti laika klases NEGLOSS
        dydis_label2 = dydis_label2 +1;
        end
      elseif strcmp(prediction(k), 'Stabdytilaika_front' ) == 1
        if score2(k,4) > slenkstis
        pozicija_lapelio2(dydis_label2,1) = posx(k,1);
        pozicija_lapelio2(dydis_label2,2) = posy(k,1);
        pozicija_lapelio2(dydis_label2,3) = wl;
        pozicija_lapelio2(dydis_label2,4) = wh;
        score_NMS2(dydis_label2,1) = score2(k,4); %taskai reikalingi NMS
        label_str2{dydis_label2} = ['Stabd_p: ' num2str(score2(k,4),'%0.2f')]; %4ame stulpeli yra
stabdyti laika klases NEGLOSS
        dydis_label2 = dydis_label2 +1;
        end
      elseif strcmp(prediction(k), 'trystaskaibandymas_back' ) == 1
        if score2(k,6) > slenkstis
        pozicija_lapelio2(dydis_label2,1) = posx(k,1);
        pozicija_lapelio2(dydis_label2,2) = posy(k,1);
        pozicija_lapelio2(dydis_label2,3) = wl;
        pozicija_lapelio2(dydis_label2,4) = wh;
        score_NMS2(dydis_label2,1) = score2(k,6); %taskai reikalingi NMS
        label_str2{dydis_label2} = ['band_n: ' num2str(score2(k,6),'%0.2f')]; %3ame stulpeli yra
stabdyti laika prazanga klases NEGLOSS
        dydis_label2 = dydis_label2 +1;
        end
    else
        %nedaryti nieko
    end

end

%dydis sumazintas 0.2 karto
for k = 1:num_predict3
    prediction(k,:) = string(predictedLabels3(k,1));
    if strcmp(prediction(k),'trystaskaibandymas_front') == 1
        if score3(k,7) > slenkstis
        pozicija_lapelio3(dydis_label3,1) = posx(k,1);
        pozicija_lapelio3(dydis_label3,2) = posy(k,1);
        pozicija_lapelio3(dydis_label3,3) = wl;
        pozicija_lapelio3(dydis_label3,4) = wh;
        score_NMS3(dydis_label3,1) = score3(k,7); %taskai reikalingi NMS
        label_str3{dydis_label3} = ['band_p: ' num2str(score3(k,7),'%0.2f')];  %5ame stulpeli yra
du taskai klases NEGLOSS
        dydis_label3 = dydis_label3 +1;
        end
     elseif strcmp(prediction(k),'Stabdlaikaprazanga_back' ) == 1
        if score3(k,1) > slenkstis
        pozicija_lapelio3(dydis_label3,1) = posx(k,1);
        pozicija_lapelio3(dydis_label3,2) = posy(k,1);
        pozicija_lapelio3(dydis_label3,3) = wl;
        pozicija_lapelio3(dydis_label3,4) = wh;
        score_NMS3(dydis_label3,1) = score3(k,1); %taskai reikalingi NMS
        label_str3{dydis_label3} = ['Prazanga_n: ' num2str(score3(k,1),'%0.2f')]; %6ame stulpeli
yra trys taskai bandymas klases NEGLOSS
        dydis_label3 = dydis_label3 +1;
        end
     elseif strcmp(prediction(k), 'Stabdlaikaprazanga_front' ) == 1
        if score3(k,2) > slenkstis
        pozicija_lapelio3(dydis_label3,1) = posx(k,1);
        pozicija_lapelio3(dydis_label3,2) = posy(k,1);
        pozicija_lapelio3(dydis_label3,3) = wl;
        pozicija_lapelio3(dydis_label3,4) = wh;
        score_NMS3(dydis_label3,1) = score3(k,2); %taskai reikalingi NMS
        label_str3{dydis_label3} = ['Prazanga_p: ' num2str(score3(k,2),'%0.2f')]; %6ame stulpeli
yra trys taskai bandymas klases NEGLOSS
        dydis_label3 = dydis_label3 +1;
        end
     elseif strcmp(prediction(k), 'Stabdlaika_back' ) == 1
        if score3(k,3) > slenkstis
        pozicija_lapelio3(dydis_label3,1) = posx(k,1);
        pozicija_lapelio3(dydis_label3,2) = posy(k,1);
        pozicija_lapelio3(dydis_label3,3) = wl;
        pozicija_lapelio3(dydis_label3,4) = wh;
        score_NMS3(dydis_label3,1) = score3(k,3); %taskai reikalingi NMS
```

```
        label_str3{dydis_label3} = ['Stabd_n: ' num2str(score3(k,3),'%0.2f')]; %4ame stulpeli yra
stabdyti laika klases NEGLOSS
        dydis_label3 = dydis_label3 +1;
        end
    elseif strcmp(prediction(k), 'Stabdlaika_front' ) == 1
        if score3(k,4) > slenkstis
        pozicija_lapelio3(dydis_label3,1) = posx(k,1);
        pozicija_lapelio3(dydis_label3,2) = posy(k,1);
        pozicija_lapelio3(dydis_label3,3) = wl;
        pozicija_lapelio3(dydis_label3,4) = wh;
        score_NMS3(dydis_label3,1) = score3(k,4); %taskai reikalingi NMS
        label_str3{dydis_label3} = ['Stabd_p: ' num2str(score3(k,4),'%0.2f')]; %4ame stulpeli yra
stabdyti laika klases NEGLOSS
        dydis_label3 = dydis_label3 +1;
        end
      elseif strcmp(prediction(k), 'trystaskaibandymas_back' ) == 1
        if score3(k,6) > slenkstis
        pozicija_lapelio3(dydis_label3,1) = posx(k,1);
        pozicija_lapelio3(dydis_label3,2) = posy(k,1);
        pozicija_lapelio3(dydis_label3,3) = wl;
        pozicija_lapelio3(dydis_label3,4) = wh;
        score_NMS3(dydis_label3,1) = score3(k,6); %taskai reikalingi NMS
        label_str3{dydis_label3} = ['band_n: ' num2str(score3(k,6),'%0.2f')]; %3ame stulpeli yra
stabdyti laika prazanga klases NEGLOSS
        dydis_label3 = dydis_label3 +1;
        end
    else
        %nedaryti nieko
    end

end

%dydis sumazintas 0.3 karto
for k = 1:num_predict4
    prediction(k,:) = string(predictedLabels4(k,1));
    if strcmp(prediction(k),'trystaskaibandymas_front') == 1
        if score4(k,7) > slenkstis
        pozicija_lapelio4(dydis_label4,1) = posx(k,1);
        pozicija_lapelio4(dydis_label4,2) = posy(k,1);
        pozicija_lapelio4(dydis_label4,3) = wl;
        pozicija_lapelio4(dydis_label4,4) = wh;
        score_NMS4(dydis_label4,1) = score4(k,7); %taskai reikalingi NMS
        label_str4{dydis_label4} = ['band_p: ' num2str(score4(k,7),'%0.2f')];  %5ame stulpeli yra
du taskai klases NEGLOSS
        dydis_label4 = dydis_label4 +1;
        end
     elseif strcmp(prediction(k),'Stabdlaikaprazanga_back' ) == 1
        if score4(k,1) > slenkstis
        pozicija_lapelio4(dydis_label4,1) = posx(k,1);
        pozicija_lapelio4(dydis_label4,2) = posy(k,1);
        pozicija_lapelio4(dydis_label4,3) = wl;
        pozicija_lapelio4(dydis_label4,4) = wh;
        score_NMS4(dydis_label4,1) = score4(k,1); %taskai reikalingi NMS
        label_str4{dydis_label4} = ['Prazanga_n: ' num2str(score4(k,1),'%0.2f')]; %6ame stulpeli
yra trys taskai bandymas klases NEGLOSS
        dydis_label4 = dydis_label4 +1;
        end
     elseif strcmp(prediction(k), 'Stabdlaikaprazanga_front' ) == 1
       if score4(k,2) > slenkstis
        pozicija_lapelio4(dydis_label4,1) = posx(k,1);
        pozicija_lapelio4(dydis_label4,2) = posy(k,1);
        pozicija_lapelio4(dydis_label4,3) = wl;
        pozicija_lapelio4(dydis_label4,4) = wh;
        score_NMS4(dydis_label4,1) = score4(k,2); %taskai reikalingi NMS
        label_str4{dydis_label4} = ['Prazanga_p: ' num2str(score4(k,2),'%0.2f')]; %6ame stulpeli
yra trys taskai bandymas klases NEGLOSS
        dydis_label4 = dydis_label4 +1;
        end
     elseif strcmp(prediction(k), 'Stabdlaika_back' ) == 1
        if score4(k,3) > slenkstis
        pozicija_lapelio4(dydis_label4,1) = posx(k,1);
        pozicija_lapelio4(dydis_label4,2) = posy(k,1);
        pozicija_lapelio4(dydis_label4,3) = wl;
        pozicija_lapelio4(dydis_label4,4) = wh;
        score_NMS4(dydis_label4,1) = score4(k,3); %taskai reikalingi NMS
        label_str4{dydis_label4} = ['Stabd_n: ' num2str(score4(k,3),'%0.2f')]; %4ame stulpeli yra
stabdyti laika klases NEGLOSS
        dydis_label4 = dydis_label4 +1;
```

```
        end
      elseif strcmp(prediction(k), 'Stabdlaika_front' ) == 1
        if score4(k,4) > slenkstis
        pozicija_lapelio4(dydis_label4,1) = posx(k,1);
        pozicija_lapelio4(dydis_label4,2) = posy(k,1);
        pozicija_lapelio4(dydis_label4,3) = wl;
        pozicija_lapelio4(dydis_label4,4) = wh;
        score_NMS4(dydis_label4,1) = score4(k,4); %taskai reikalingi NMS
        label_str4{dydis_label4} = ['Stabd_p: ' num2str(score4(k,4),'%0.2f')]; %4ame stulpeli yra
stabdyti laika klases NEGLOSS
        dydis_label4 = dydis_label4 +1;
        end
      elseif strcmp(prediction(k), 'trystaskaibandymas_back' ) == 1
        if score4(k,6) > slenkstis
        pozicija_lapelio4(dydis_label4,1) = posx(k,1);
        pozicija_lapelio4(dydis_label4,2) = posy(k,1);
        pozicija_lapelio4(dydis_label4,3) = wl;
        pozicija_lapelio4(dydis_label4,4) = wh;
        score_NMS4(dydis_label4,1) = score4(k,6); %taskai reikalingi NMS
        label_str4{dydis_label4} = ['band_n: ' num2str(score4(k,6),'%0.2f')]; %3ame stulpeli yra
stabdyti laika prazanga klases NEGLOSS
        dydis_label4 = dydis_label4 +1;
        end
    else
        %nedaryti nieko
    end

end

%dydis sumazintas 0.4 karto
for k = 1:num_predict5
    prediction(k,:) = string(predictedLabels5(k,1));
    if strcmp(prediction(k),'trystaskaibandymas_front') == 1
        if score5(k,7) > slenkstis
        pozicija_lapelio5(dydis_label5,1) = posx(k,1);
        pozicija_lapelio5(dydis_label5,2) = posy(k,1);
        pozicija_lapelio5(dydis_label5,3) = wl;
        pozicija_lapelio5(dydis_label5,4) = wh;
        score_NMS5(dydis_label5,1) = score5(k,7); %taskai reikalingi NMS
        label_str5{dydis_label5} = ['band_p: ' num2str(score5(k,7),'%0.2f')];  %5ame stulpeli yra
du taskai klases NEGLOSS
        dydis_label5 = dydis_label5 +1;
        end
     elseif strcmp(prediction(k),'Stabdlaikaprazanga_back' ) == 1
        if score5(k,1) > slenkstis
        pozicija_lapelio5(dydis_label5,1) = posx(k,1);
        pozicija_lapelio5(dydis_label5,2) = posy(k,1);
        pozicija_lapelio5(dydis_label5,3) = wl;
        pozicija_lapelio5(dydis_label5,4) = wh;
        score_NMS5(dydis_label5,1) = score5(k,1); %taskai reikalingi NMS
        label_str5{dydis_label5} = ['Prazanga_n: ' num2str(score5(k,1),'%0.2f')]; %6ame stulpeli
yra trys taskai bandymas klases NEGLOSS
        dydis_label5 = dydis_label5 +1;
        end
     elseif strcmp(prediction(k), 'Stabdlaikaprazanga_front' ) == 1
        if score5(k,2) > slenkstis
        pozicija_lapelio5(dydis_label5,1) = posx(k,1);
        pozicija_lapelio5(dydis_label5,2) = posy(k,1);
        pozicija_lapelio5(dydis_label5,3) = wl;
        pozicija_lapelio5(dydis_label5,4) = wh;
        score_NMS5(dydis_label5,1) = score5(k,2); %taskai reikalingi NMS
        label_str5{dydis_label5} = ['Prazanga_p: ' num2str(score5(k,2),'%0.2f')]; %6ame stulpeli
yra trys taskai bandymas klases NEGLOSS
        dydis_label5 = dydis_label5 +1;
        end
     elseif strcmp(prediction(k), 'Stabdytilaika_back' ) == 1
        if score5(k,3) > slenkstis
        pozicija_lapelio5(dydis_label5,1) = posx(k,1);
        pozicija_lapelio5(dydis_label5,2) = posy(k,1);
        pozicija_lapelio5(dydis_label5,3) = wl;
        pozicija_lapelio5(dydis_label5,4) = wh;
        score_NMS5(dydis_label5,1) = score5(k,3); %taskai reikalingi NMS
        label_str5{dydis_label5} = ['Stabd_n: ' num2str(score5(k,3),'%0.2f')]; %4ame stulpeli yra
stabdyti laika klases NEGLOSS
        dydis_label5 = dydis_label5 +1;
        end
      elseif strcmp(prediction(k), 'Stabdytilaika_front' ) == 1
        if score5(k,4) > slenkstis
```

```
        pozicija_lapelio5(dydis_label5,1) = posx(k,1);
        pozicija_lapelio5(dydis_label5,2) = posy(k,1);
        pozicija_lapelio5(dydis_label5,3) = wl;
        pozicija_lapelio5(dydis_label5,4) = wh;
        score_NMS5(dydis_label5,1) = score5(k,4); %taskai reikalingi NMS
        label_str5{dydis_label5} = ['Stabd_p: ' num2str(score5(k,4),'%0.2f')]; %4ame stulpeli yra
stabdyti laika klases NEGLOSS
        dydis_label5 = dydis_label5 +1;
        end
      elseif strcmp(prediction(k), 'trystaskaibandymas_back' ) == 1
        if score5(k,6) > slenkstis
        pozicija_lapelio5(dydis_label5,1) = posx(k,1);
        pozicija_lapelio5(dydis_label5,2) = posy(k,1);
        pozicija_lapelio5(dydis_label5,3) = wl;
        pozicija_lapelio5(dydis_label5,4) = wh;
        score_NMS5(dydis_label5,1) = score5(k,6); %taskai reikalingi NMS
        label_str5{dydis_label5} = ['band_n: ' num2str(score5(k,6),'%0.2f')]; %3ame stulpeli yra
stabdyti laika prazanga klases NEGLOSS
        dydis_label5 = dydis_label5 +1;
        end
    else
        %nedaryti nieko
    end

end

%dydis sumazintas 0.5 karto
for k = 1:num_predict6
    prediction(k,:) = string(predictedLabels6(k,1));
    if strcmp(prediction(k),'trystaskaibandymas_front') == 1
        if score6(k,7) > slenkstis
        pozicija_lapelio6(dydis_label6,1) = posx(k,1);
        pozicija_lapelio6(dydis_label6,2) = posy(k,1);
        pozicija_lapelio6(dydis_label6,3) = wl;
        pozicija_lapelio6(dydis_label6,4) = wh;
        score_NMS6(dydis_label6,1) = score6(k,7); %taskai reikalingi NMS
        label_str6{dydis_label6} = ['band_p: ' num2str(score6(k,7),'%0.2f')];  %5ame stulpeli yra
du taskai klases NEGLOSS
        dydis_label6 = dydis_label6 +1;
        end
    elseif strcmp(prediction(k),'Stabdlaikaprazanga_back' ) == 1
        if score6(k,1) > slenkstis
        pozicija_lapelio6(dydis_label6,1) = posx(k,1);
        pozicija_lapelio6(dydis_label6,2) = posy(k,1);
        pozicija_lapelio6(dydis_label6,3) = wl;
        pozicija_lapelio6(dydis_label6,4) = wh;
        score_NMS6(dydis_label6,1) = score6(k,1); %taskai reikalingi NMS
        label_str6{dydis_label6} = ['Prazanga_n: ' num2str(score6(k,1),'%0.2f')]; %6ame stulpeli
yra trys taskai bandymas klases NEGLOSS
        dydis_label6 = dydis_label6 +1;
        end
     elseif strcmp(prediction(k), 'Stabdlaikaprazanga_front' ) == 1
        if score6(k,2) > slenkstis
        pozicija_lapelio6(dydis_label6,1) = posx(k,1);
        pozicija_lapelio6(dydis_label6,2) = posy(k,1);
        pozicija_lapelio6(dydis_label6,3) = wl;
        pozicija_lapelio6(dydis_label6,4) = wh;
        score_NMS6(dydis_label6,1) = score6(k,2); %taskai reikalingi NMS
        label_str6{dydis_label6} = ['Prazanga_p: ' num2str(score6(k,2),'%0.2f')]; %6ame stulpeli
yra trys taskai bandymas klases NEGLOSS
        dydis_label6 = dydis_label6 +1;
        end
     elseif strcmp(prediction(k), 'Stabdytilaika_back' ) == 1
        if score6(k,3) > slenkstis
        pozicija_lapelio6(dydis_label6,1) = posx(k,1);
        pozicija_lapelio6(dydis_label6,2) = posy(k,1);
        pozicija_lapelio6(dydis_label6,3) = wl;
        pozicija_lapelio6(dydis_label6,4) = wh;
        score_NMS6(dydis_label6,1) = score6(k,3); %taskai reikalingi NMS
        label_str6{dydis_label6} = ['Stabd_n: ' num2str(score6(k,3),'%0.2f')]; %4ame stulpeli yra
stabdyti laika klases NEGLOSS
        dydis_label6 = dydis_label6 +1;
        end
      elseif strcmp(prediction(k), 'Stabdytilaika_front' ) == 1
        if score6(k,4) > slenkstis
        pozicija_lapelio6(dydis_label6,1) = posx(k,1);
        pozicija_lapelio6(dydis_label6,2) = posy(k,1);
        pozicija_lapelio6(dydis_label6,3) = wl;
```

```
        pozicija_lapelio6(dydis_label6,4) = wh;
        score_NMS6(dydis_label6,1) = score6(k,4); %taskai reikalingi NMS
        label_str6{dydis_label6} = ['Stabd_p: ' num2str(score6(k,4),'%0.2f')]; %4ame stulpeli yra
stabdyti laika klases NEGLOSS
        dydis_label6 = dydis_label6 +1;
        end
      elseif strcmp(prediction(k), 'trystaskaibandymas_back' ) == 1
        if score6(k,6) > slenkstis
        pozicija_lapelio6(dydis_label6,1) = posx(k,1);
        pozicija_lapelio6(dydis_label6,2) = posy(k,1);
        pozicija_lapelio6(dydis_label6,3) = wl;
        pozicija_lapelio6(dydis_label6,4) = wh;
        score_NMS6(dydis_label6,1) = score6(k,6); %taskai reikalingi NMS
        label_str6{dydis_label6} = ['band_n: ' num2str(score6(k,6),'%0.2f')]; %3ame stulpeli yra
stabdyti laika prazanga klases NEGLOSS
        dydis_label6 = dydis_label6 +1;
        end
    else
        %nedaryti nieko
    end

end

%% Parodyti vaizda su lapeliais
testImage = imread('F:\TEST_DATA_NAUDOTI\Stabdytilaika_back\test2.jpg');

%du taskai komandos label
output_image = insertObjectAnnotation(testImage,'rectangle', pozicija_lapelio, label_str);
output_image2 = insertObjectAnnotation(img_test2,'rectangle', pozicija_lapelio2, label_str2);
output_image3 = insertObjectAnnotation(img_test3,'rectangle', pozicija_lapelio3, label_str3);
output_image4 = insertObjectAnnotation(img_test4,'rectangle', pozicija_lapelio4, label_str4);
output_image5 = insertObjectAnnotation(img_test5,'rectangle', pozicija_lapelio5, label_str5);
output_image6 = insertObjectAnnotation(img_test6,'rectangle', pozicija_lapelio6, label_str6);

figure
imshow(output_image);
figure
imshow(output_image2);
figure
imshow(output_image3);
figure
imshow(output_image4);
figure
imshow(output_image5);
figure
imshow(output_image6);

% NMS
[pozicija_1,score_1,index_1] = selectStrongestBbox(pozicija_lapelio,score_NMS1,
'OverlapThreshold', 0.3);
[pozicija_2,score_2,index_2] = selectStrongestBbox(pozicija_lapelio2,score_NMS2,
'OverlapThreshold', 0.3);
[pozicija_3,score_3,index_3] = selectStrongestBbox(pozicija_lapelio3,score_NMS3,
'OverlapThreshold', 0.3);
[pozicija_4,score_4,index_4] = selectStrongestBbox(pozicija_lapelio4,score_NMS4,
'OverlapThreshold', 0.3);
[pozicija_5,score_5,index_5] = selectStrongestBbox(pozicija_lapelio5,score_NMS5,
'OverlapThreshold', 0.3);
[pozicija_6,score_6,index_6] = selectStrongestBbox(pozicija_lapelio6,score_NMS6,
'OverlapThreshold', 0.3);


komanda1 = 'Prazanga_p';
komanda2 = 'Prazanga_n';
komanda3 = 'Stabd_n';
komanda4 = 'Stabd_p';
komanda5 = 'band_n';
komanda6 = 'band_p';

%lapeliu priskyrimas NMS
count1 = numel(index_1);
count2 = numel(index_2);
count3 = numel(index_3);
count4 = numel(index_4);
count5 = numel(index_5);
count6 = numel(index_6);

label_dydis_NMS1 = 1;
```

```
label_dydis_NMS2 = 1;
label_dydis_NMS3 = 1;
label_dydis_NMS4 = 1;
label_dydis_NMS5 = 1;
label_dydis_NMS6 = 1;

%NMS vizualizacija
for g = 1: count1
   index_tikrinti = index_1(g,1);
    if contains(label_str{index_tikrinti}, komanda1)
        label_str_NMS1{label_dydis_NMS1 } = ['praz_n: ' num2str(score_1(g,1),'%0.2f')];
        label_dydis_NMS1 = label_dydis_NMS1 + 1;
    elseif contains(label_str{index_tikrinti}, komanda2)
        label_str_NMS1{label_dydis_NMS1 } = ['praz_p: ' num2str(score_1(g,1),'%0.2f')];
        label_dydis_NMS1 = label_dydis_NMS1 + 1;
    elseif contains(label_str{index_tikrinti}, komanda3)
        label_str_NMS1{label_dydis_NMS1 } = ['Stabd_n: ' num2str(score_1(g,1),'%0.2f')];
        label_dydis_NMS1 = label_dydis_NMS1 + 1;
    elseif contains(label_str{index_tikrinti}, komanda4)
         label_str_NMS1{label_dydis_NMS1 } = ['Stabd_p: ' num2str(score_1(g,1),'%0.2f')];
         label_dydis_NMS1 = label_dydis_NMS1 + 1;
    elseif contains(label_str{index_tikrinti}, komanda5)
         label_str_NMS1{label_dydis_NMS1 } = ['band_n: ' num2str(score_1(g,1),'%0.2f')];
         label_dydis_NMS1 = label_dydis_NMS1 + 1;
    elseif contains(label_str{index_tikrinti}, komanda6)
         label_str_NMS1{label_dydis_NMS1 } = ['band_p: ' num2str(score_1(g,1),'%0.2f')];
         label_dydis_NMS1 = label_dydis_NMS1 + 1;
    end
end

for g = 1: count2
   index_tikrinti = index_2(g,1);
    if contains(label_str2{index_tikrinti}, komanda1)
        label_str_NMS2{label_dydis_NMS2 } = ['praz_n: ' num2str(score_2(g,1),'%0.2f')];
        label_dydis_NMS2 = label_dydis_NMS2 + 1;
    elseif contains(label_str2{index_tikrinti}, komanda2)
        label_str_NMS2{label_dydis_NMS2 } = ['praz_p: ' num2str(score_2(g,1),'%0.2f')];
        label_dydis_NMS2 = label_dydis_NMS2 + 1;
    elseif contains(label_str2{index_tikrinti}, komanda3)
        label_str_NMS2{label_dydis_NMS2 } = ['Stabd_n: ' num2str(score_2(g,1),'%0.2f')];
        label_dydis_NMS2 = label_dydis_NMS2 + 1;
    elseif contains(label_str2{index_tikrinti}, komanda4)
         label_str_NMS2{label_dydis_NMS2 } = ['Stabd_p: ' num2str(score_2(g,1),'%0.2f')];
         label_dydis_NMS2 = label_dydis_NMS2 + 1;
    elseif contains(label_str2{index_tikrinti}, komanda5)
         label_str_NMS2{label_dydis_NMS2 } = ['band_n: ' num2str(score_2(g,1),'%0.2f')];
         label_dydis_NMS2 = label_dydis_NMS2 + 1;
    elseif contains(label_str2{index_tikrinti}, komanda6)
         label_str_NMS2{label_dydis_NMS2 } = ['band_p: ' num2str(score_2(g,1),'%0.2f')];
         label_dydis_NMS2 = label_dydis_NMS2 + 1;
    end
end

for g = 1: count3
   index_tikrinti = index_3(g,1);
    if contains(label_str3{index_tikrinti}, komanda1)
        label_str_NMS3{label_dydis_NMS3 } = ['praz_n: ' num2str(score_3(g,1),'%0.2f')];
        label_dydis_NMS3 = label_dydis_NMS3 + 1;
    elseif contains(label_str3{index_tikrinti}, komanda2)
        label_str_NMS3{label_dydis_NMS3 } = ['praz_p: ' num2str(score_3(g,1),'%0.2f')];
        label_dydis_NMS3 = label_dydis_NMS3 + 1;
    elseif contains(label_str3{index_tikrinti}, komanda3)
        label_str_NMS3{label_dydis_NMS3 } = ['Stabd_n: ' num2str(score_3(g,1),'%0.2f')];
        label_dydis_NMS3 = label_dydis_NMS3 + 1;
    elseif contains(label_str3{index_tikrinti}, komanda4)
         label_str_NMS3{label_dydis_NMS3 } = ['Stabd_p: ' num2str(score_3(g,1),'%0.2f')];
         label_dydis_NMS3 = label_dydis_NMS3 + 1;
    elseif contains(label_str3{index_tikrinti}, komanda5)
         label_str_NMS3{label_dydis_NMS3 } = ['band_n: ' num2str(score_3(g,1),'%0.2f')];
         label_dydis_NMS3 = label_dydis_NMS3 + 1;
    elseif contains(label_str3{index_tikrinti}, komanda6)
         label_str_NMS3{label_dydis_NMS3 } = ['band_p: ' num2str(score_3(g,1),'%0.2f')];
         label_dydis_NMS3 = label_dydis_NMS3 + 1;
    end
end

for g = 1: count4
   index_tikrinti = index_4(g,1);
```

```
    if contains(label_str4{index_tikrinti}, komanda1)
        label_str_NMS4{label_dydis_NMS4 } = ['praz_n: ' num2str(score_4(g,1),'%0.2f')];
        label_dydis_NMS4 = label_dydis_NMS4 + 1;
    elseif contains(label_str4{index_tikrinti}, komanda2)
        label_str_NMS4{label_dydis_NMS4 } = ['praz_p: ' num2str(score_4(g,1),'%0.2f')];
        label_dydis_NMS4 = label_dydis_NMS4 + 1;
    elseif contains(label_str4{index_tikrinti}, komanda3)
        label_str_NMS4{label_dydis_NMS4 } = ['Stabd_n: ' num2str(score_4(g,1),'%0.2f')];
        label_dydis_NMS4 = label_dydis_NMS4 + 1;
    elseif contains(label_str4{index_tikrinti}, komanda4)
         label_str_NMS4{label_dydis_NMS4 } = ['Stabd_p: ' num2str(score_4(g,1),'%0.2f')];
         label_dydis_NMS4 = label_dydis_NMS4 + 1;
    elseif contains(label_str4{index_tikrinti}, komanda5)
         label_str_NMS4{label_dydis_NMS4 } = ['band_n: ' num2str(score_4(g,1),'%0.2f')];
         label_dydis_NMS4 = label_dydis_NMS4 + 1;
    elseif contains(label_str4{index_tikrinti}, komanda6)
         label_str_NMS4{label_dydis_NMS4 } = ['band_p: ' num2str(score_4(g,1),'%0.2f')];
         label_dydis_NMS4 = label_dydis_NMS4 + 1;
    end
end

for g = 1: count5
   index_tikrinti = index_5(g,1);
    if contains(label_str5{index_tikrinti}, komanda1)
        label_str_NMS5{label_dydis_NMS5 } = ['praz_n: ' num2str(score_5(g,1),'%0.2f')];
        label_dydis_NMS5 = label_dydis_NMS5 + 1;
    elseif contains(label_str5{index_tikrinti}, komanda2)
        label_str_NMS5{label_dydis_NMS5 } = ['praz_p: ' num2str(score_5(g,1),'%0.2f')];
        label_dydis_NMS5 = label_dydis_NMS5 + 1;
    elseif contains(label_str5{index_tikrinti}, komanda3)
        label_str_NMS5{label_dydis_NMS5 } = ['Stabd_n: ' num2str(score_5(g,1),'%0.2f')];
        label_dydis_NMS5 = label_dydis_NMS5 + 1;
    elseif contains(label_str5{index_tikrinti}, komanda4)
         label_str_NMS5{label_dydis_NMS5 } = ['Stabd_p: ' num2str(score_5(g,1),'%0.2f')];
         label_dydis_NMS5 = label_dydis_NMS5 + 1;
    elseif contains(label_str5{index_tikrinti}, komanda5)
         label_str_NMS5{label_dydis_NMS5 } = ['band_n: ' num2str(score_5(g,1),'%0.2f')];
         label_dydis_NMS5 = label_dydis_NMS5 + 1;
    elseif contains(label_str5{index_tikrinti}, komanda6)
         label_str_NMS5{label_dydis_NMS5 } = ['band_p: ' num2str(score_5(g,1),'%0.2f')];
         label_dydis_NMS5 = label_dydis_NMS5 + 1;
    end
end

for g = 1: count6
   index_tikrinti = index_6(g,1);
    if contains(label_str6{index_tikrinti}, komanda1)
        label_str_NMS6{label_dydis_NMS6 } = ['praz_n: ' num2str(score_6(g,1),'%0.2f')];
        label_dydis_NMS6 = label_dydis_NMS6 + 1;
    elseif contains(label_str6{index_tikrinti}, komanda2)
        label_str_NMS6{label_dydis_NMS6 } = ['praz_p: ' num2str(score_6(g,1),'%0.2f')];
        label_dydis_NMS6 = label_dydis_NMS6 + 1;
    elseif contains(label_str6{index_tikrinti}, komanda3)
        label_str_NMS6{label_dydis_NMS6 } = ['Stabd_n: ' num2str(score_6(g,1),'%0.2f')];
        label_dydis_NMS6 = label_dydis_NMS6 + 1;
    elseif contains(label_str6{index_tikrinti}, komanda4)
         label_str_NMS6{label_dydis_NMS6 } = ['Stabd_p: ' num2str(score_6(g,1),'%0.2f')];
         label_dydis_NMS6 = label_dydis_NMS6 + 1;
    elseif contains(label_str6{index_tikrinti}, komanda5)
         label_str_NMS6{label_dydis_NMS6 } = ['band_n: ' num2str(score_6(g,1),'%0.2f')];
         label_dydis_NMS6 = label_dydis_NMS6 + 1;
    elseif contains(label_str6{index_tikrinti}, komanda6)
         label_str_NMS6{label_dydis_NMS6 } = ['band_p: ' num2str(score_6(g,1),'%0.2f')];
         label_dydis_NMS6 = label_dydis_NMS6 + 1;
    end
end

%Vizualizacija NMS
output_image_NMS1 = insertObjectAnnotation(testImage,'rectangle', pozicija_1, label_str_NMS1);
output_image_NMS2 = insertObjectAnnotation(testImage,'rectangle', pozicija_2, label_str_NMS2);
output_image_NMS3 = insertObjectAnnotation(testImage,'rectangle', pozicija_3, label_str_NMS3);
output_image_NMS4 = insertObjectAnnotation(testImage,'rectangle', pozicija_4, label_str_NMS4);
output_image_NMS5 = insertObjectAnnotation(testImage,'rectangle', pozicija_5, label_str_NMS5);
output_image_NMS6 = insertObjectAnnotation(testImage,'rectangle', pozicija_6, label_str_NMS6);

figure
imshow(output_image_NMS1);
figure
```

```
imshow(output_image_NMS2);
figure
imshow(output_image_NMS3);
figure
imshow(output_image_NMS4);
figure
imshow(output_image_NMS5);
figure
imshow(output_image_NMS6);
```

Priedas nr. 2, Matlab .m programinis kodas, RCNN detektorius

```
% %% Clear data
   clear all;
   close all;
   clc;

%% Uzkrauti training data ir ivertinti vieno paveikslelio dydi
Gestai   = fullfile('D:\TesejoGestaiOriginal_Sorted_NAUDOTI_ISKIRPTI _LBP_100_75');

Treniravimo_duomenys = imageDatastore(Gestai,  'IncludeSubfolders', true, 'LabelSource',
'foldernames');
Testavimo_duomenys = imageDatastore(Gestai,  'IncludeSubfolders', true, 'LabelSource',
'foldernames');

img = readimage(Treniravimo_duomenys,550);
img = rgb2gray(img);
img = histeq(img);

lbp_bruozai = extractLBPFeatures(img, 'Cellsize', [4 4]);


lbpFeatureSize = length(lbp_bruozai);
%isskiriu is visu duomenu LBP bruozus
numImages = numel(Treniravimo_duomenys.Files);
trainingFeatures = zeros(numImages, lbpFeatureSize, 'single');

for i = 1:numImages
    img = readimage(Treniravimo_duomenys, i);

    img = rgb2gray(img);
    img = histeq(img);

    trainingFeatures(i, :) = extractLBPFeatures(img, 'Cellsize', [4 4]);
end

%lapeliai kiekvienai klaisei
treniravimo_lapeliai = Treniravimo_duomenys.Labels;

%% TREE klasifikatorius
classifier = TreeBagger(100, trainingFeatures, treniravimo_lapeliai, 'Method','classification');

%isskiriu is visu duomenu LBP bruozus
numImages = numel(Testavimo_duomenys.Files);
testFeatures = zeros(numImages, lbpFeatureSize, 'single');

for i = 1:numImages
    img = readimage(Testavimo_duomenys, i);

    img = rgb2gray(img);
    img = histeq(img);

    testFeatures(i, :) = extractLBPFeatures(img, 'Cellsize', [4 4]);
end
%lapeliai kiekvienai klaisei
testavimo_lapeliai = Testavimo_duomenys.Labels;

% Make class predictions using the test features.
predictedLabels = predict(classifier, testFeatures);
predictedLabels = grp2idx(predictedLabels);
predictedLabels = predictedLabels.';

% konvertuojam lapelius i MxN matricas
num_sample = numel(testavimo_lapeliai);
```

```
outputs = zeros(7,num_sample);
outputsIdx = sub2ind(size(outputs), predictedLabels, 1:num_sample);
outputs(outputsIdx) = 1;

%test data
treniravimo_lapeliai = grp2idx(treniravimo_lapeliai);
treniravimo_lapeliai = treniravimo_lapeliai.';
targets = zeros(7,num_sample);

targetsIdx = sub2ind(size(targets), treniravimo_lapeliai, 1:num_sample);
targets(targetsIdx) = 1;


figure;
plotconfusion(targets,outputs);
```

kodas 5

```
%% Clear data
      clear all;
      close all;
      clc;

%neuro tinklo parametrai

layers = [imageInputLayer([32 32 3]);
          convolution2dLayer(5,20);
          reluLayer();
          maxPooling2dLayer(2,'Stride',2);
          fullyConnectedLayer(7); %kiek bus klasiu output
          softmaxLayer();
          classificationLayer()];

   % Set training options
 options = trainingOptions('sgdm', ...
        'MiniBatchSize', 128, ...
        'InitialLearnRate', 1e-8 , ...
        'LearnRateSchedule', 'piecewise', ...
        'LearnRateDropFactor', 0.1, ...
        'LearnRateDropPeriod', 100, ...
        'MaxEpochs', 20, ...
        'Verbose', true, ...
        'Plots','training-progress');

%treniruoti tinkla
tic
detector = trainFasterRCNNObjectDetector(komandos, layers, options);
toc

tic
img = imread('F:\TEST_DATA_NAUDOTI\Stabdytilaika_back\test1.jpg');
%Run detector.

[bboxes,score,label] = detect(detector,img);

% Display the detection results
[score, idx] = max(score);

%aptikti komandas
bbox = bboxes(idx, :);
annotation = sprintf('%s: (Tikimybe = %f)', label(idx), score);

outputImage = insertObjectAnnotation(img, 'rectangle', bbox, annotation);

figure
imshow(outputImage)

save rcnn_class_500 detector
```