VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS INSTITUTAS
KOMPIUTERINIO IR DUOMENŲ MODELIAVIMO KATEDRA

Magistro baigiamasis darbas

Biologinių membranų defektų atpažinimo metodai atominės jėgos mikroskopo nuotraukose

Atliko:
Tautminas Cibulskis parašas

Vadovas:
dr. Tadas Meškauskas

Vilnius
2023

Turinys

Sutartinis terminų žodynas

- AJM (angl. Atomic Force Microscopy, trumpinys AFM) – atominės jėgos mikroskopija.
- KNT (angl. Convolutional Neural Network, trumpinys CNN) – konvoliucinis neuroninis tinklas.
- YOLO (angl. You Only Look Once) – konvoliucinio neuroninio tinklo modelis.
- TP (angl. true positive) – teisingai rastų defektų kiekis.
- FP (angl. false positive) – neteisingai rastų defektų kiekis.
- FN (angl. false negative) – nerastų defektų kiekis.
- IoU (angl. intersection over union) – sankirtos ir sąjungos santykis.
- mAP (angl. mean average precision) – vidutinių tikslumų vidurkis.

Santrauka

Šiuo darbu siekiama spręsti defektų aptikimo uždavinį biologinių membranų atominės jėgos mikroskopijos nuotraukose. Darbe pateikiama biologinės membranos, atominės jėgos mikroskopijos, skaitmeninio vaizdo apdorojimo algoritmų, defektų nustatymo metodų ir objektų aptikimo algoritmų įvertinimo teorijos apžvalga. Aprašoma defektų aptikimo ne dirbtinio intelekto metodais (Hough apskritimų transformacijos algoritmu, Suzuki kontūrų radimo algoritmu) ir įvarios architektūros konvoliuciniais neuroniniais tinklais (YOLOv4, CenterNet HourGlass104 512x512, EfficientDet D0 512x512, SSD ResNet152 V1 FPN 640x640, Faster R-CNN Inception ResNet V2 640x640) susijusi teorija. Darbe analizuojami PASCAL VOC 2012 mAP, MS COCO 2014 mAP, painiavos matricos, precision, recall, F-1-score ir vidutinio IoU metrikų rezultatai gauti programiškai įgyvendinus septynis paminėtuosius defektų aptikimo metodus.

Nagrinėti įvairios architektūros konvoliucinių neuroninių tinklų metodai yra reikšmingai pranašesni už metodus, kurie nesinaudoja dirbtiniu intelektu. Geriausią PASCAL VOC 2012 mAP rezultatą pasiekė YOLOv4 modelis su 72,88%. MS COCO 2014 mAP metrikoje geriausią rezultatą parodė CenterNet HourGlass104 512x512 modelis su 27,20%. Sprendžiant defektų aptikimo uždavinį biologinių membranų atominės jėgos mikroskopijos nuotraukose PASCAL VOC 2012 mAP metrikos rezultatas yra svarbesnis už MS COCO 2014 mAP. Dėl to darbe rekomenduojamas defektų aptikimas YOLOv4 modeliu. Bandymai pagerinti YOLOv4 konvoliucinio neuroninio tinklo gaunamas metrikas pridedant papildomą skaitmeninio vaizdo apdorojimo metodą (išbandyti Canny kraštų aptikimo, Otsu ir Sobel briaunų radimo operatoriaus metodai) buvo nesėkmingi.

Summary

Methods for Detection of Biological Membrane Defects in Atomic Force Microscopy Images

A biological membrane is a separating layer which forms the outer boundary of a living cell. It is responsible for vital functions such as protecting the cell, regulating the transport of substances in and out of the cell, receiving chemical messengers from other cells and acting as a receptor. In nature, biological membranes can be damaged by toxins which cause the formation of defects on the membrane and due to this reason toxins are often the pathogens of dangerous diseases. Automated defect detection can help diagnose formidable diseases caused by toxins faster. One of the ways to analyse biological membranes is images from atomic force microscopy. Automated defect detection problem could be solved by applying object detection methods on atomic force microscopy images.

This paper is composed of six chapters. In chapter 1 a review of related scientific literature is provided. Chapter 2 consists of theory related to the biological membrane, defects of biological membranes and atomic force microscopy. Chapter 3 has the theory of image preprocessing. Chapter 4 discusses the theory related to the methods of defect detection which are applied in this research. Chapter 5 provides a theory of metrics for the object detection task. Chapter 6 analyses the practical part of the research and discusses the results. At the end of the paper plans for further work are provided.

The task of detection of defects in atomic microscopy images was carried out by using non-artificial intelligence methods (Hough circle transformation algorithm, Suzuki contour finding algorithm) and convolutional neural networks of various architectures (YOLOv4, CenterNet HourGlass104 512x512, EfficientDet D0 512x512, SSD ResNet152 V1 FPN 640x640, Faster R-CNN Inception ResNet V2 640x640). Data was prepared with augmentation and resizing for convolutional neural networks. All aforementioned methods were evaluated with the PASCAL VOC 2012 mAP and MS COCO 2014 metrics. YOLOv4 reached the highest value in PASCAL VOC 2012 mAP with 72,88% and CenterNet HourGlass104 512x512 reached the highest value in MS COCO mAP with 27,2%. When solving the problem of defect detection in atomic force microscopy images of biological membranes, the result of the PASCAL VOC 2012 mAP metric is more important than the MS COCO 2014 mAP. Due to that, the usage of the YOLOv4 model is recommended. Attempts to improve the metrics obtained by the YOLOv4 convolutional neural network by adding a digital image preprocessing method (Canny edge detection, Otsu and Sobel edge detection operator methods were tested) were unsuccessful.

Įvadas

Plazminė membrana yra kiekvieno gyvo organizmo ląstelę dengianti organelė. Membrana ne tik atskiria ląstelę nuo aplinkos taip apsaugodama ją nuo išorinių veiksnių, bet ir atlieka kitas gyvybiškai svarbias funkcijas kaip kad ląstelės formos palaikymas ar dalyvavimas medžiagų transportavime. Plazminė membrana yra sudaryta iš baltymų ir lipidų. Į membraną įsiterpiančios medžiagos gali sudaryti porą taip sukurdamos membranos defektą.

Naudojantis į membraną įsiterpiančias medžiagas membranose defektai gali būti sukelti tikslingai. Gamtoje šios į membraną įsiterpiančios ir porų defektus sudarančios medžiagos yra išskiriami bakterijų baltymai vadinami toksinais. Nepriklausomai nuo toksinų šaltinio veikimo principas išlieka toks pat. Dėl defektų membranose sukėlimo toksinai yra dažnai pavojingų ligų patogenai. Taip atsiranda **defektų aptikimo problema**. Automatizuotas toksinų aptikimas gali pasitarnauti visuomenei greičiau diagnozuojant toksinų sukeltas ligas. Taip pat neautomatizuotas defektų žymėjimas yra laikui reiklus darbas su žmogiškųjų klaidų faktoriumi dėl potencialiai didelio defektų kiekio.

Vienas iš būdų tyrinėti plazmines membranas ir defektus jose yra AJM metodu gauti vaizdai. Atominės jėgos mikroskopas turi elektroninę sistemą, kuri judina antgalį (ilgis paprastai būna mažesnis nei μm) per medžiagos paviršių. Į antgalį yra nukreiptas lazeris, kuris atsimušęs nuo antgalio į detektorių suteikia sistemai informaciją apie tyrinėjamos medžiagos paviršių. Mikroskopo elektroninė sistema gautą informaciją apdoroja skirtingais būdais, todėl rezultatai gaunami iš skirtingų kanalų.

Šiame tyrime dirbama su Vilniaus universiteto Gyvybės mokslų centro suteiktais duomenimis. Duomenys sudaryti iš biologinių membranų vaizdų, kurie gaunami pasinaudojant AJM. Vaizduose matomi defektai t. y. fosfolipidų dvisluoksnio pažeidimai. Magistro baigiamajame darbe studijuojami būdai gerinti vaizduose matomų defektų detekcijas.

Išskiriami du **darbo tikslai**:

1. Metodikų automatizuotai biologinių membranų defektų detekcijai palyginimas.
2. Išankstinių vaizdo apdorojimo metodų įtakos defektų detekcijos metrikoms palyginimas.

Siekiant įgyvendinti darbo tikslus buvo atliktas defektų aptikimas ne dirbtinio intelekto metodais (Hough apskritimų transformacijos algoritmu, Suzuki kontūrų radimo algoritmu) ir įvairios architektūros konvoliuciniais neuroniniais tinklais (YOLOv4, CenterNet HourGlass104 512x512, EfficientDet D0 512x512, SSD ResNet152 V1 FPN 640x640, Faster R-CNN Inception ResNet V2 640x640). Metodai buvo įvertinti PASCAL VOC 2012 mAP, MS COCO 2014 mAP, painiavos matricos, precision, recall, F-1-score ir vidutinio IoU metrikų rezultatais. Programavimo darbų aplinkos aprašymas pateiktas 6.2 skyriuje. PASCAL VOC 2012 mAP metrikoje geriausią rezultatą pasiekė YOLOv4 modelis su 72,88%, o MS COCO 2014 mAP metrikoje CenterNet HourGlass104 512x512 KNT su 27,20%. Darbe argumentuojamas PASCAL VOC 2012 mAP metrikos pranašumas prieš MS COCO 2014 mAP šio tyrimo kontekste. Dėl to rekomenduojamas YOLOv4 modelio naudojimas sprendžiant defektų aptikimo problemą AJM nuotraukose. Atlikti eksperimentai siekiant pagerinti YOLOv4 konvoliucinio neuroninio tinklo gaunamas metrikas pridedant papildomą skaitmeninio vaizdo apdorojimo metodą (išbandyti Canny kraštų aptikimo, Otsu ir Sobel briaunų radimo operatoriaus metodai) rezultatų nepagerino.

Magistro baigiamojo **darbo struktūra**:

- 1 skyrius. Pateikiama susijusių mokslinių darbų apžvalga.

- 2 skyrius. Pateikiama su biologine membrana, biologinės membranos defektais ir atominės jėgos mikroskopija susijusi teorija.
- 3 skyrius. Pateikiamas skaitmeninio vaizdo apibrėžimas ir teoriškai aptariami tyrimo metu atlikti skaitmeninio vaizdo apdorojimo algoritmai.
- 4 skyrius. Pateikiama tyrimo metu atliktų defektų aptikimo atominės jėgos mikroskopijos skaitmeniniuose vaizduose metodų teorija.
- 5 skyrius. Pateikiama objektų aptikimo skaitmeniniuose vaizduose metodų įvertinimo teorija.
- 6 skyrius. Pateikiamas tyrimo metu atliktų praktinių darbų ir jų rezultatų aprašas.

Šis darbas yra mokslo tiriamojo darbo tęsinys. Iš darbo autoriaus mokslo tiriamojo darbo [18] **paimtos dalys** (visi naudoti skyriai šiame darbe buvo modifikuoti):

- Susijusių darbų apžvalgos skyrius, kuriame buvo aprašytas Hessiano Lašelių algoritmas (angl. Hessian Blob Algorithm), Tiny YOLO konvoliucinis neuroninis tinklas kraujo kūnelių aptikimui, Faster R-CNN konvoliucinio neuroninio pritaikymas vėžinių ląstelių aptikimui ir aprašyti buvusių magistrantų šios problemos sprendimo metodai su rezultatais.
- Išankstinio vaizdų apdorojimo metodų skyrius su trumpu duomenų aprašymu. Šiame skyriuje atlikta mažiausiai pakeitimų.
- Defektų aptikimo metodų teorijos skyrius su Hough apskritimų transformacijos, Suzuki algoritmo ir konvoliucinio neuroninio tinklo teorija.

1. Susijusių darbų apžvalga

Kompiuterinės regos moksliniai darbai, kurie analizuoja objektų aptikimo skaitmeniniuose vaizduose uždavinį, gali būti išskirti į dvi atskiras kategorijas: prieš ir po 2014 metus, kuomet atsirado giliojo mokymosi metodai. Viola-Jones (2001 m.) objektų aptikimo karkasas yra pamatinis darbas nuo kurio prasidėjo dabartinių tradicinių objektų aptikimo metodų tobulinimas. 2006 metais buvo pristatytas HOG (angl. histogram of oriented gradients) požymių aptikimo vaizde metodas. 2010 metais aprašytas DPM (angl. deformable part model) objektų aptikimo metodas, kuris naudojasi HOG ir pristatė ribojančių stačiakampių (angl. bounding box) regresijos idėją. Nuo 2014 metų atsiradę giliojo mokymosi objektų aptikimo modeliai klasifikuojami į dvi kategorijas: vieno etapo (angl. one-stage) ir dviejų etapų (angl. two-stage) modeliai. Šios kategorijos skiriasi tuo, kad dviejų etapų kategorijos modeliai pirmajame etape bando aptikti regionus, kuriuose potencialiai bus objektai ir tik tuomet atlikti jų aptikimą, o vieno etapo modeliai bando aptikti objektus iš karto iš įvesties duomenų neatliekant regionų nustatymo. Dviejų etapų objektų aptikimo modeliai: RCNN (2014 m.), SPPNet (2014 m.), Fast RCNN (2015 m.), Faster RCNN (2015 m.), Mask R-CNN (2017 m.), Feature Pyramid Network/FPN (2017 m.), G-RCNN (2021 m.). Vieno etapo objektų aptikimo modeliai: YOLO (2016 m.), SSD (2016 m.), RetinaNet (2017 m.), YOLOv3 (2018 m.), YOLOv4 (2020 m.), YOLOR (2021 m.).

2018 m. B. Marsh ir kiti autoriai pasiūlė metodą defektų atpažinimo AJM nuotraukose automatizavimui vadinamą Hessiano Lašelių algoritmu (angl. Hessian Blob Algorithm) [27]. Straipsnyje įrodyti algortimo objektų aptikimo tikslumo ir atsparumo triukšmui pranašumai prieš anksčiau konvencinius dalelių aptikimo AJM Threshold ir Watershed algoritmus (programinį įgyvendinimą galima rasti Gwyddion programinėje įrangoje). Hessiano Lašelių algoritmas priešingai nei Threshold ir Watershed algoritmai nereikalauja išankstinio vaizdo apdorojimo ir algoritmo parametrų. 2021 m. J. G. Beton [3] pristatė su Pyhon programavimo kalba sukurtą atviro kodo įrankį Topostats, kuris skirtas automatizuotam biomolekulių aptikimui AJM nuotraukose. 2022 m. Vilniaus universiteto mokslininkai [31] palygino defektų aptikimo problemos sprendimą TopoStats programa su KNT. Topostats programa davė reikšmingai prastesnius rezultatus $recall$ metrikoje nei KNT, t. y. reikšminga dalis defektų su Topostats buvo neaptikta.

Kita dalis mokslinių darbų analizuojančių objektų aptikimą AJM nuotraukose naudojasi giliojo mokymosi modeliais. 2021 m. H. Bai kartu su S.Wu [2] pasiūlė modelį, kuris aptinka nano vielas (angl. nanowire) AJM. Modelis susideda iš trijų dalių: YOLOv3 aptinka nano vielą ir ją pažymi ribojančiu stačiakampiu, stačiakampyje atliekamos morfologinės operacijos, BI-LSTM-CRF modelis segmentuoja vielą kaip liniją. YOLOv3 su 70% slenksčiu įvertinant ar ribojantis stačiakampis yra geras pasiekė 98.4% tikslumą ir 95.5% $recall$. 2021 m. J. Sotres [39] pateikė tos pačios molekulės atpažinimo skirtinguose AJM vaizduose problemos sprendimo pasiūlymą. Problema spręsta pritaikius YOLOv3 objektų aptikimo modelį ir Siamese modelį, kuris identifikavo tą pačią molekulę skirtinguose vaizduose. YOLOv3 su tyrimo duomenimis pasiekė 91% mAP.

Giliojo mokymosi modeliai naudojami ir kitokių mikroskopijos tipų skaitmeninių vaizdų tyrimuose. 2016 m. J. Zhang ir kiti mokslininkai pasiūlė metodą vėžinių ląstelių aptikimui fazės kontrasto mikroskopijos (angl. phase-contrast microscopy) nuotraukose naudojant Faster R-CNN KNT kartu su apskritimų skenavimo algoritmu (angl. circle scanning algorithm). Apskritimų skenavimo algoritmas naudojamas po KNT tam, kad būtų aptiktos dėl adhezijos sukibusios ląstelės. Pasiūlytas metodas pasiekė 97,4% tikslumą, 96,1% $recall$, 90,7% AP ir 92,4% AUC [51]. 2019 m. M. M. Alam ir M. T. Islam sprendė kraujo ląstelių automatinį aptikimą ir jų kiekio nustatymą su Nikon ECLIPSE 50i mikroskopu gautuose vaizduose. Pasiūlytas problemos sprendimas pasi-

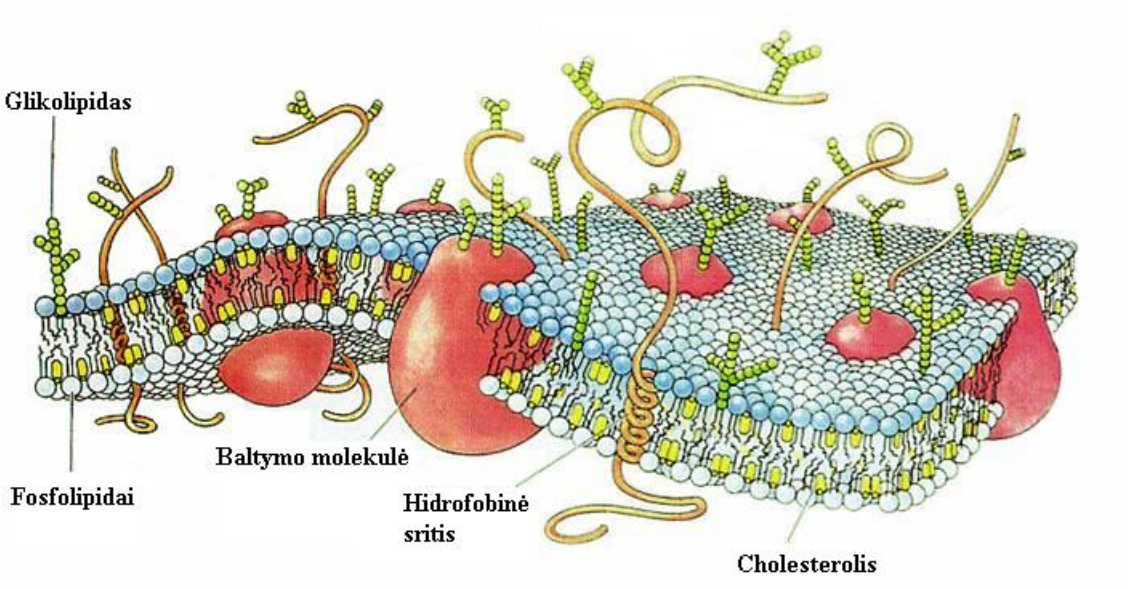
naudojant Tiny YOLO KNT. Didžiausias objektų aptikimo tikslumas pagal klasę gavosi 96.36%, o blogiausias 86.89%, mAP reikšmė tyrime buvo lygi 62.36% [1].

Biologinių membranų defektų aptikimo problemą AJM nuotraukose yra sprendę buvę fakulteto magistrantai. Tyrimuose buvo naudojamasi tais pačiais duomenimis kaip ir šiame darbe. Magistro studijų baigiamąjį darbą 2020 m. Vilniaus universitete apgynė Vytenis Navalinskas darbo tema "Defektų atpažinimas biologinėse membranose" [28]. Darbe objektų aptikimas atliktas šablonų atitikimo, Hough apskritimų transformacijos ir apmokyto Tiny YOLOv3 architektūros KNT metodais. Metodai buvo įvertinti tikslumo (angl. accuracy) metrika. Šablonų atitikimo metodas pasiekė 5.1% tikslumą, Hough apskritimų transformacija 23.3%, o apmokytas Tiny YOLOv3 KNT davė 26.3% tikslumą. 2021m. Matematikos ir informatikos fakulteto magistro studijas pabaigė Igor Vilghelm darbo tema "Automatizuoti biologinių membranų defektų atpažinimo AJM nuotraukose" [47]. Darbe objektų aptikimas atliktas Hough apskritimų transformacijos ir apmokyto Tiny YOLOv4 architektūros KNT metodais. Analizuotas Tiny YOLOv4 architektūros KNT pasiekė $\sim$88% tikslumą ir 52.28% mAP.

2. Biologinių membranų atominės jėgos mikroskopijos teorija

2.1. Biologinė membrana

Biologinė membrana (literatūroje dar galimai vadinama plazmine membrana arba ląstelės membrana) yra organelė, kuri gaubia gyvūnų ir augalų ląstelę ir atskiria ją nuo išorinės aplinkos [40]. Biologinės membranų storis varijuoja nuo 5 iki 10 nm (nano yra SI sistemos priešdėlis, kuris reiškia daugiklį 10^{-9}). Biologinės membranos yra sudarytos iš lipidų dvisluoksnio su baltymais dvisluoksnio viduje (žr. pav. 1). Tarp skirtingų ląstelių šie komponentai varijuoja, nes skirtingos ląstelės vykdo skirtingus procesus ir joms reikalingi specifiški baltymai ir specifiški fosfolipidai. Biologinės membranos atlieka izoliacijos nuo supančios aplinkos, signalų atpažinimo naudojant receptorius, medžiagų transporto, apsaugos nuo aplinkos sąlygų pasikeitimų, selektyvus medžiagų pralaidumo, formos suteikimo, katalizines funkcijas. Biologinių membranų analizė yra potencialus būdas sustabdyti mirtinas ligas [48].



1 pav. Biologinės membranos sandaros iliustracija [46].

Septynioliktame amžiuje sukūrus mikroskopą buvo išsiaiškinta, kad organizmai yra sudaryti iš ląstelių. Devynioliktame amžiuje buvo suprasta, kad ląstelę gaubia barjeras, kuris ir yra biologinė membrana. Dvidešimto amžiaus viduryje sukūrus elektroninį mikroskopą (angl. electron microscope) išaiškėjo iš ko sudaryta membrana. Dabartinis biologinę membraną apibūdinantis membranos modelis buvo pasiūlytas 1972 m., jį sukūrė mokslininkai Singer ir Nicholson, jis dar vadinamas skysčių mozaikos modeliu (angl. fluid mosaic model).

Visi membranos lipidai yra amfipatiniai, t. y. lipidų molekulės turi ir hidrofilinę (sąveikaujančią su vandeniu), ir hidrofobinę (nesąveikaujančią su vandeniu) dalis. Amfipatiškumas leidžia selektyviai praleisti medžiagas. Lipidai yra išskiriami į tris klases:

- Fosfolipidai. Šį lipidų klasė yra pagrindinė biologinių membranų sudedamoji dalis. Fosfolipidų molekulės sudarytos iš glicerolio, riebalų rūgščių ir fosfato grupės. Riebalų rūgštys yra hidrofobinės, o fosfatas hidrofilinis. Fosfolipidai sudaro dvisluoksnį, kur hidrofilinės galvutės nukreiptos į ląstelės vidaus ir išorės skysčius, o hidrofobinės uodegėlės į membranos vidų.

- Glikolipidai. Ši lipidų klasė sudaryta iš glicerolio pagrindo, prie kurio prijungta viena ar daugiau cukraus molekulių. Eukariotinėse plazminėse membranose sudaro apie 5% visų lipidų molekulių.

- Steroliai (cholesterolis). Gyvūnų ląstelių plazminėse membranose sterolių klasės lipidai sudaro apie 20% visų lipidų. Ši lipidų klasė sudaryta iš hidroksilo grupės, keturių žiedų steroido struktūros ir trumpos angliavandenilių šoninės grandinės.

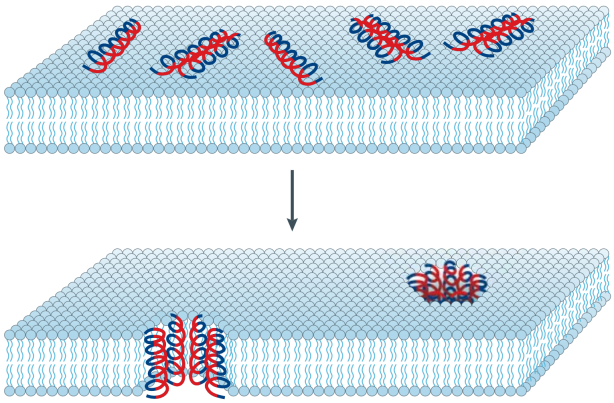
Baltymų molekulių membranose yra mažiau nei fosfolipidų, tačiau jos yra didesnės už fosfolipidų molekules, todėl baltymai sudaro apie pusę visos biologinės membranos masės. Nepaisant to, kad baltymai nėra kertinė biologinės membranos struktūros dalis jie atlieka svarbias katalizavimo, transportavimo ir atpažinimo funkcijas. Pagal baltymų jungtį su plazmine membrana išskiriamos dvi jų klasės:

- Integraliniai (angl. intrinsic) baltymai. Šie baltymai yra visiškai įsiterpę į membraną arba kerta jos dalį.

- Periferiniai (angl. extrinsic) baltymai. Šie baltymai yra už membranos. Dažniausiai būdami už membranos jie būna prikibę prie kitų integralinių baltymų. Ši baltymų klasė svarbi signalų persiuntimui.

2.2. Biologinės membranos defektai

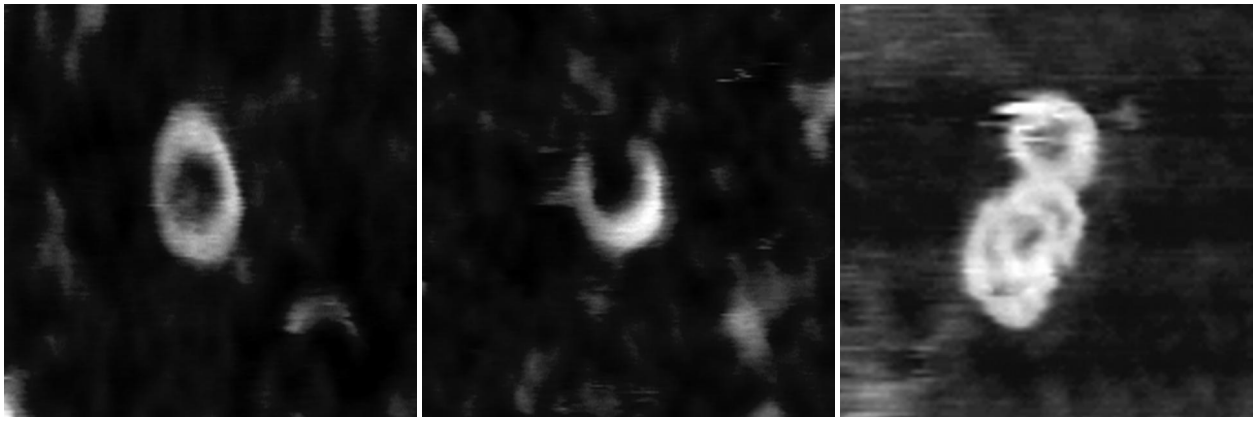
Biologinėje membranoje defektai (literatūroje dar galimai vadinami ląstelės sienelės poromis) atsiranda dėl mechaninių arba cheminių veiksnių [20]. Natūraliai atsirandantys defektai yra dažni, o jų tankis priklauso nuo membranos formavimosi sąlygų [43]. Dėl lipidinės membranos savybės savaime susiformuoti, tokie defektai greitai išnyksta. Tačiau aplinkoje pasitaiko medžiagų, dažnai bakterijų išskiriamų toksinų, membranoje formuojančių poras ir taip pažeidžiančių ląstelę. Veikiant šiems toksinams, ląstelės netenka savo pagrindinio barjero ir dėl to žūsta. Šių patogenų sukeliamos ligos dažnai yra pavojingos ir jas būtina greitai diagnozuoti. Tokių medžiagų detekcija yra labai svarbi ankstyvai ligų diagnostikai. Defektus membranoje galima sukelti taip pat ir dirbtiniu būdu tikslingai naudojant įvairias į membraną įsiterpiančias medžiagas. Šios medžiagos patekusios ant membranos paviršiaus, jungiasi tarpusavyje ir taip suformuoja membranoje porą (žr. pav. 2). Membranoje atsiradęs defektas gali būti apibūdinamas ritinio formos skyle tarp fosfolipidų.



2 pav. Defektų susidarymas biologinėje membranoje ant jos patekus poras formuojantiems peptidams [14].

Dėl membraną veikiančių skirtingų veiksnių gali susidaryti skirtingų tipų defektai biologinėse membranose:

- Pilno dydžio apskritimai. Retkarčiais šio tipo defektai būna ir netaisyklingos apskritimo formos.
- Arkos formos defektai, nepilno apskritimo defektai. Galimi 1/4, 1/2, 3/4 apskritimo defektai. Retkarčiais šio tipo defektai būna ir netaisyklingos apskritimo dalies formos.
- Defektų klasteriai (telkiniai). Defektai šiame tipe vienas kitą dengia, t. y. jie susiformuoja vienas ant kito. Didelė defektų koncentracija tam tikrame plote. Minimaliai trys persidengiantys defektai.



(a) Pilno dydžio apskritimas. (b) Arkos formos defektas. (c) Defektų klasteris.

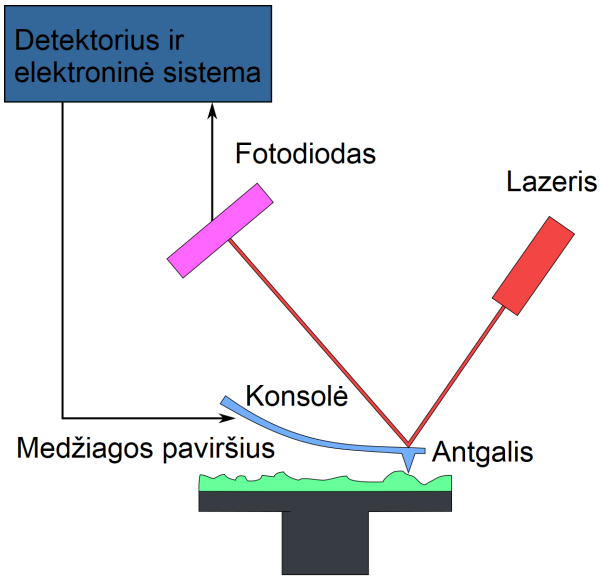
3 pav. Skirtingų defektų tipų biologinėse membranose pavyzdžiai aukščio kanalo vaizduose.

2.3. Atominės jėgos mikroskopija

AJM skenuoja medžiagų paviršius nuo nanometrinės (nano yra SI sistemos priešdėlis, kuris reiškia daugiklį 10^{-9}) skalės iki mikrometrinės skalės (mikro yra SI sistemos priešdėlis, kuris reiškia daugiklį 10^{-6}). Vertikalus mikroskopo diapazonas standartiškai yra 8–10 μm. AJM renka

informacija apie paviršių per jį braukdami mažą antgalį. Supaprastintas veikimo principo pavyzdys galėtų būti žmogaus informacijos rinkimas apie paviršių per jį braukiant pirštą. AJM buvo sukurtas mokslininkų Gerd Binning, Calvin Quate ir Christoph Gerber 1985 metais [12], po metų už šį atradimą suteiktas Nobelio prizas fizikos srityje.

AJM (žr. pav. 4) per medžiagą braukia konsolę, kuri turi aštrų antgalį (30nm spindulio) priekyje. Kuomet konsolė atsiduria pakankamai arti paviršiaus ji pradedama veikti paviršiuje esančių jėgų. Tai priverčia antgalį pajudėti ir dėl to konsolė susilenkia. Konsolės lenkiamieji judesiai yra fiksuojami lazerio ir fotodetektoriaus. Fotodetektorius konvertuoja šviesą į elektrinį signalą. Skenavimas vyksta linijomis, paviršius nuskenuojamas linija į vieną pusę ir tuomet grįžtama į pradinį tašką, konsolė paslenkama statmenai į šoną tik ką skenuotai linijai ir procesas kartojamas iš naujo. Įprastai skenavimo duomenys yra įrašomi kaip nespalvotas vaizdas, šio tyrimo duomenys irgi yra nespalvoti.



4 pav. Atominės jėgos mikroskopo sandaros iliustracija [45].

AJM gali analizuoti standžių paviršių mėginius esančius tiek ore, tiek įmerktus į skystį. AJM gebėjimas tirti mėginius įmerktus į skystį yra viena iš priežasčių kodėl šio tipo mikroskopai sėkmingai taikomi biomedicininės krypties moksluose [13]. Puikus to pavyzdys yra galimybė analizuoti ląstelių paviršius, kurios yra įmerktos į standartinę Petri lėkštę. Priklausomai nuo tyrime naudojamo AJM elektroninė sistema iš fotodetektoriaus gautus elektrinių signalų duomenis apdoroja skirtingais metodais. Dėl šios priežasties AJM vaizdų duomenys yra gaunami iš skirtingų kanalų. Kanalų kiekis priklauso nuo naudojamo mikroskopo. Šio tyrimo duomenys yra trijų kanalų: aukščio, fazės ir amplitudės (žr. priedą A).

3. Išankstinio vaizdų apdorojimo metodai

3.1. Skaitmeninis vaizdas

Vaizdas yra vizualinė objekto reprezentacija. Skaitmeninis vaizdas yra funkcija $f(x, y)$, kur x bei y yra baigtinės, diskrečios reikšmės ir x, y reprezentuoja vaizdo pikselį, funkcija laiko pikselio intensyvumo reikšmę [42]. Matematiškai skaitmeninis vaizdas yra dvidimencinio vaizdo reprezentavimas matrica, kurioje yra baigtinis skaičius elementų (jie ir vadinami pikseliais). Kiekvienas pikselis turi skaitinę reikšmę. Kai kurie pikseliai gali turėti daugiau nei vieną reikšmę skirtinguose pikselio kanaluose. Įprastai pikselių reikšmės yra reprezentuojamos 8 bitais, t. y. pakliūna į intervalą [0; 255].

Priklausomai nuo pikselį apibrėžiančių reikšmių skaitmeniniai vaizdai yra išskiriami į kategorijas [21]:

- Binarinis vaizdas. Pikseliai laiko 1 bito informaciją. Skaitmeninis vaizdas yra sudarytas iš juodos (reprezentuojamos 0) ir baltos (reprezentuojamos 1) spalvų.
- Pilko atspalvio vaizdas. Pikseliai laiko 8 bitų informaciją. Skaitmeninis vaizdas yra sudarytas iš pilkų atspalvių, kurie identifikuoja spalvos intensyvumą. Čia juoda reprezentuojama 0, o balta 255 pikselio reikšme.
- Spalvotas vaizdas. Pikseliai turi tris kanalus, kiekvienas iš jų laiko 8 bitų informaciją. Pikselių kanalai vadinami RGB (angl. **R**ed, **G**reen, **B**lue), kur kanalas R – raudonos spalvos intensyvumas, G – žalios spalvos intensyvumas ir B – mėlynos spalvos intensyvumo reprezentavimas. Čia juoda spalva reprezentuojama visų kanalų 0 reikšmėmis, o balta visų kanalų 255 reikšmėmis.

Ne taip plačiai paplitusiuose skaitmeninių vaizdų formatuose galimas informacijos laikymas kitokiame bitų kiekyje ar kitoks kanalų skaičius. Pavyzdžiui, spalvotų skaitmeninių vaizdų formatas RGBA (angl. **R**ed, **G**reen, **B**lue, **A**lpha) be spalvų intensyvumo kanalų turi ketvirtąjį – permatomumo kanalą. Taip pat yra ir kitokių spalvotų vaizdų formatų kaip kad HSL, HWB, CMYK ar NCol. Šiame darbe bus nagrinėjamos anksčiau apibrėžtos plačiai paplitusios trys skaitmeninių vaizdų grupės.

3.2. Objektų sužymėjimas

Formatai, kuriais galima sužymėti skaitmeninio vaizdo objektus: Coco JSON, Pascal VOC XML, Tensorflow TFRecord, YOLO Darknet TXT, CreateML JSON, Yolov7 Pytorch TXT, Yolov5 Pytorch TXT, LabelMr JSON, Google CLoud AutoML Vision CSV, SuperAnnotate JSON, Sagemaker GroundTryth Manifest, Supervisely JSON, Scale AI JSON, LabelBox JSON, VGG Image Annotator JSON, VGG Image Annotator CSV, VoTT JSON, VoTT CSV, LabelBox Video JSON, IBM Cloud Annotations JSON, Scaled-YOLOv4 TXT, Unity Perception JSON, OIDv4 TXT, YOLO Keras TXT, OpenAI CLIP Classification, RetinaNet Keras CSV, OpenImages CSV, Tensorflow Object Detection CSV, Muticlass Classification CSV, YOLOv4 PyTorch TXT, Kaggle Wheat CSV, Udacity TXT, Marmot XML, YOLOv5 Oriented Bounding Boxes.

Tyrime dirbta su Pascal VOC XML, Tensorflow TFRecord, YOLO Darknet TXT ir Tensorflow Object Detection CSV skaitmeninių vaizdų žymėjimais. Pascal VOC yra XML formato failas, šiame objektų žymėjimo formate sukuriamas atskiras failas kiekvienam vaizdui. Šis formatas yra

dažnas objektų žymėjimo vaizduose informacijos apsikeitime, tačiau modeliai tiesiogiai nenaudoja Pascal VOC XML formato failo mokymuisi. Tensorflow TFRecord formate sukuriamas vienas binarinis failas visai tam tikra paskirtimi naudojamai duomenų ir meta duomenų aibei. Šis formatas pasižymi mažu atminties užėmimu, greitomis įvesties ir išvesties operacijomis dirbant su TensorFlow Python programavimo kalbos biblioteka. YOLO Darknet TXT formate sukuriamas atskiras failas kiekvienam vaizdui. Defektas faile pažymimas atskira eilute, kuri prasideda klasės numeriu ir normalizuotomis koordinatėmis intervale [0;1], skaičiai atskiriami tarpais. Šiuo formatu naudojasi Darknet karkasas, kuris turi programiškai įgyvendinęs skirtingas YOLO modelių versijas. Tensorflow Object Detection CSV formate sukuriamas vienas failas visai duomenų aibei. Tensorflow Object Detection CSV formatas yra tarpinis formatas pavertimui į Tensorflow TFRecord formatą. Tensorflow Object Detection API dirba su Tensorflow TFRecord formatu, bet jis yra binarinis failas, kuris neperskaitomas žmogui, todėl iš pradžių sukuriamas Tensorflow Object Detection CSV formatas, o iš jo Tensorflow TFRecord formatas. Naudojamų duomenų formatų pavyzdžiai yra pateikti šio darbo prieduose (žr. priedą C).

3.3. Spalvoto vaizdo konvertavimas į nespalvotą vaizdą

Verčiant spalvotą vaizdą į nespalvotą yra prarandama dalis informacijos. Negalima spalvotą vaizdą paversti į nespalvotą ir tada atgal į spalvotą neprarandant vaizdo kokybės. Spalvotas vaizdas turis tris kanalus kiekvienam pikseliui, o nespalvotas vaizdas kiekvienam pikseliui turi vieną kanalą. Tarkime, kad R – nagrinėjamo pikselio raudonos spalvos kanalo reikšmė, G – žalios ir B – mėlynos. Tuomet nagrinėjamo pikselio pilkos spalvos intensyvumas Y bus gaunamas pasinaudojus šia formule [11]:

$$Y = R * 0.299 + G * 0.587 + B * 0.114 \tag{3.1}$$

Tai nėra vienintelis būdas gauti nespalvotą vaizdą. Formulėje esantys koeficientai kai kur yra pakeičiami.

3.4. Konvoliucija ir kryžminė koreliacija

Konvoliucija yra svarbi operacija skaitmeninių vaizdų apdorojimo srityje, kurios panaudojimas išskiria tam tikrus vaizdo bruožus. Vieni iš daugelio konvoliucijos operacijos panaudojimo būdų pavyzdžių yra vaizdo ryškinimas (angl. sharpening), glodinimas (angl. blurring) ir kraštų aptikimas (angl. edge detection). Konvoliucijos operacija naudoja filtrus (angl. kernel) – matricas. Paprastai tariant konvoliucija yra dviejų matricų (skaitmeninio vaizdo pikselio reikšmių matricos ir filtro matricos) elementų daugybos paeiliui suma, kuomet filtro matrica prieš operaciją yra apsukama 180 laipsnių. Konvoliucijos operacijos matricos turi tokį patį dimensijų skaičių.

Formaliai matematikoje konvoliucija tarp dviejų funkcijų $f, g : \mathbb{R}^d \to \mathbb{R}$ apibrėžiama šia formule:

$$(f * g)(\mathbf{x}) = \int f(\mathbf{z})g(\mathbf{x} - \mathbf{z})d\mathbf{z} \tag{3.2}$$

Kadangi dirbama su skaitmeniniais vaizdais, t. y. su diskrečiu objektu, integralas tampa diskrečių reikšmių suma. Vienos dimensijos diskreti konvoliucija apibrėžiama taip:

$$(f * g)(i) = \sum_a f(a)g(i - a) \tag{3.3}$$

Tarkime, kad dirbame su skaitmeniniu vaizdu I ir turime dviejų dimensijų filtrą K. Tuomet esame pasiruošę apibrėžti dviejų dimensijų diskrečios konvoliucijos operaciją konkrečiam vaizdui ir filtrui [23]:

$$(I * K)(i, j) = \sum_m \sum_n I(m, n)K(i - m, j - n) \tag{3.4}$$

Dėl komutatyvumo savybės 3.4 formulė gali būti perrašyti į šią išraiška:

$$(I * K)(i, j) = \sum_m \sum_n I(i - m, j - n)K(m, n) \tag{3.5}$$

Komutatyvumo savybė atsirado dėl to, nes kaip jau šiame skyriuje buvo minėta filtro matrica apsukama 180 laipsnių. Vienintelė priežastis apsukti matricą ir yra komutatyvumas. Komutatyvumo savybė yra naudinga dėl formulės 3.5, kurią yra patogiau programiškai įgyvendinti mašininiame mokyme nei 3.4 formulę. Dažnai komutatyvumas nėra svarbi savybė neuroniniams tinklams, todėl dalis neuroninių tinklų vietoj konvoliucijos naudoja kryžminės koreliacijos funkciją. Kryžminė koreliacija yra tas pats kaip konvoliucija, bet be filtro apsukimo:

$$(I * K)(i, j) = \sum_m \sum_n I(i + m, j + n)K(m, n) \tag{3.6}$$

3.5. Slenkstinis triukšmo filtras

Slenkstinis triukšmo filtravimas yra juodai balto vaizdo sudarymo procesas iš pilko atspalvio vaizdo, kuomet tam tikros nuo filtro priklausančios pikselių intensyvumo reikšmės nulemia ar pikselis bus baltas, ar juodas. Dėl savo paprastumo ir intuityvių savybių slenkstinis triukšmo filtras yra svarbus vaizdų segmentavimo algoritmas. Tarkime turime skaitmeninį vaizdą $f(x, y)$, kurį sudaro šviesūs objektai tamsiame fone, taip kad sudarius vaizdo pikselių intensyvumo histogramą turėsime du pikus. Tuomet objektus galima segmentuoti pasirinkus slenkstį T, kuris atskiria histogramoje esančius pikus. Bet koks skaitmeninio vaizdo pikselis (x, y), kurio intensyvumas $f(x, y) \geq T$ bus laikomas objektų tašku, o likę taškai bus fonas [22]. Toks segmentavimo būdas vadinamas viršutiniu slenksčiu. Iš viso yra keturi slenkstinio triukšmo filtro tipai pagal filtro naudojimą (viršutinis, apatinis, išorinis, vidinis) [38]. Visi slenkstinio triukšmo filtro tipai remiasi ta pačia idėja, kaip ir aprašytas atvejis, bet filtrą naudoja skirtingai.

Formaliai segmentuotas vaizdas $g(x, y)$, gaunamas pasinaudojus slenkstiniu triukšmo filtru priklausomai nuo naudoto filtro tipo, apibrėžiamas šiuo būdu:

- Viršutinis slenkstinis filtras (dar vadinamas binariniu filtru):

$$g(x, y) = \begin{cases} 1, & \text{jeigu } f(x, y) \geq T \\ 0, & \text{jeigu } f(x, y) < T. \end{cases} \tag{3.7}$$

- Apatinis slenkstinis filtras (dar vadinamas atvirkštiniu binariniu filtru):

$$g(x, y) = \begin{cases} 1, & \text{jeigu } f(x, y) \leq T \\ 0, & \text{jeigu } f(x, y) > T. \end{cases} \tag{3.8}$$

- Išorinis slenkstinis filtras:

$$g(x,y) = \begin{cases} 1, & \text{jeigu } f(x,y) < T_1 \text{ arba } f(x,y) > T_2 \\ 0, & \text{jeigu } T_1 \leq f(x,y) \leq T_2. \end{cases} \tag{3.9}$$

- Vidinis slenkstinis filtras:

$$g(x,y) = \begin{cases} 1, & \text{jeigu } T_1 \leq f(x,y) \leq T_2 \\ 0, & \text{jeigu } f(x,y) < T_1 \text{ arba } f(x,y) > T_2. \end{cases} \tag{3.10}$$

Yra ir daugiau slenkstinio triukšmo filtro išskiriamų tipų [24] [22]. Kuomet pasirinktas slenkstis T yra pastovus visam skaitmeniniam vaizdui, tuomet tai vadinama globaliu slenksčiu. Atvejais, kai T reikšmė keičiasi filtruojant skaitmeninį vaizdą, naudojamas terminas kintamas slenkstis. Jeigu pasirinktas kintamas slenkstis T bet kuriame skaitmeninio vaizdo pikselyje (x, y) yra priklausomas nuo kaimyninių (x, y) pikselių, tada slenkstinis filtravimas vadinamas lokaliu, o jeigu T priklauso nuo pačios (x, y) intensyvumo reikšmės, tuomet slenkstinis filtravimas vadinamas dinaminiu.

3.6. Otsu metodas

Dažniausiai taikomas slenkstinis triukšmo filtras yra viršutinis globalus slenkstinis triukšmo filtras. Pagrindinis klausimas, susijęs su slenkstinio triukšmo filtravimu, yra, kokia turėtų būti pasirinkta slenksčio T reikšmė. Otsu metodas automatizuotai parenka slenksčio reikšmę maksimizuodamas tarpklasinę dispersiją. Otsu algoritmas daro prielaidą, kad skaitmeninis vaizdas turi dvi pikselių klases (vaizdo objektų ir fono) ir iš to susidarančią bimodalinę pikselių intensyvumo histogramą. Otsu metodas automatizuoja slenksčio parinkimą atsižvelgdamas į dispersiją, bet yra algoritmų, kurių slenksčio automatizavimas remiasi entropija ar tikimybių teorija [19].

Tarkime, kad L žymi pilko intensyvumo reikšmių skaičių $[1, 2, ..., L]$. Pikselių skaičius su intensyvumo reikšme i žymimas n_i, taigi bendras pikselių skaičius yra $N = n_1 + n_2 + ... + n_L$. Toliau formalizuojant Otsu metodą skaitmeninio vaizdo pilkų atspalvių histograma normalizuojama ir traktuojama kaip tikimybinis pasiskirstymas:

$$p_i = n_i/N, \quad p_i \geq 0, \quad \sum_{i=1}^{L} p_i = 1 \tag{3.11}$$

Tuomet pritaikant viršutinį globalų slenkstinį triukšmo filtrą su reikšme k, skaitmeninio vaizdo pikseliai bus išskaidyti į dvi klases (fono ir objektų) C_0 ir C_1. C_0 žymi pikselius $[1, ..., k]$ ir C_1 žymi pikselius $[k + 1, ..., L]$. Įsivedus žymėjimus galime apsibrėžti klasių atsitikimo tikimybes (atitinkamai ω_0 ir ω_1) ir jų intensyvumo vidurkius (atitinkamai μ_0 ir μ_1):

$$\omega_0 = Pr(C_0) = \sum_{i=1}^{k} p_i \tag{3.12}$$

$$\omega_1 = Pr(C_1) = \sum_{i=k+1}^{L} p_i \tag{3.13}$$

$$\mu_0 = \sum_{i=1}^{k} iPr(i|C_0) = \sum_{i=1}^{k} ip_i/\omega_0 \tag{3.14}$$

$$\mu_1 = \sum_{i=k+1}^{L} iPr(i|C_1) = \sum_{i=k+1}^{L} ip_i/\omega_1 \tag{3.15}$$

Metodo kūrėjas, mokslininkas Nobuyuki Otsu, nagrinėjo bendrą dispersiją, tarpklasinę dispersiją ir atskirų klasių dispersiją bei šių dispersijų santykius [29]. Ieškant optimalaus slenksčio k^* galima apsiriboti tarpklasinės dispersijos maksimizavimu. Tarpklasinė dispersija su įsivestomis sąvokomis apibrėžiama šiuo būdu:

$$\sigma_B^2 = \omega_0\omega_1(\mu_1 - \mu_0)^2 \tag{3.16}$$

Naudodamiesi 3.16 formule randame optimalų slenkstį k^* maksimizuodami σ_B^2. Maksimali reikšmė visada egzistuoja. Formalus k^* apibrėžimas:

$$\sigma_B^2(k^*) = \max_{1\leq k<L} \sigma_B^2(k) \tag{3.17}$$

3.7. Gauso glodinimo filtras

Gauso glodinimo filtras yra 2D konvoliucinis operatorius (matrica), kuris naudojamas kompiuterinių vaizdų glodinimui. Vaizdo glodinimo idėja tokia pati kaip ir vidurkio filtro, skirtumas tik tas, kad skiriasi matricos reikšmės. Gauso glodinimo matricos reikšmės atitinka Gauso pasiskirstymo funkcijos varpo formą. Vienmatės erdvės Gauso pasiskirstymo funkcija matoma 3.18. Čia σ yra pasiskirstymo standartinis nuokrypis. Pasiskirstymo vidurkio reikšmė lygi nuliui. Dvimatės erdvės Gauso pasiskirstymo funkcija matoma 3.19.

$$G(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-x^2}{2\sigma^2}} \tag{3.18}$$

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{\frac{-(x^2+y^2)}{2\sigma^2}} \tag{3.19}$$

Kadangi kompiuterinis vaizdas yra laikomas diskrečių pikselių pavidalu, prieš atliekant konvoliuciją reikia atlikti Gauso funkcijos diskrečią aproksimaciją. Teoriškai Gauso pasiskirstymo reikšmės nėra lygios nuliui, todėl turėtų būti be galo didelė konvoliucijos matrica, bet praktiškai Gauso pasiskirstymo reikšmės nutolusios per daugiau nei 3 standartinius nuokrypius nuo vidurkio yra labai arti nulio, todėl jas galima tokiomis ir laikyti, apibrėžiant matricos dydį. Vienareikšmiškai sutarto metodo aproksimuoti Gauso funkciją nėra.

Suradus konvoliucijai naudojamą matricą konvoliucijos operaciją galima atlikti greičiau nei įprastai atliekant viendimenses horizontalias ir vertikalias konvoliucijas. 2D konvoliucija atliekama iš pradžių įgyvendinant 1D konvoliuciją x kryptimi ir tada atliekant 1D konvoliuciją y kryptimi. Glodinimo lygis priklauso nuo standartinio nuokrypio σ reikšmės. Gauso glodinimo operacijos išvestis yra kiekvieno pikselio kaimynų svorinis vidurkis, kurio svoris yra didesnis centriniuose pikseliuose. Dėl to Gauso filtravimas duoda švelnesnį glodinimą nei vidurkio glodinimas ir geriau išsaugo paveikslėlyje esančias briaunas nei vidurkio glodinimas [44].

3.8. Sobel briaunų radimo operatorius

Sobel operatorius literatūroje dar vadinamas Sobel–Feldman kraštų aptikimo filtru. Šio operatoriaus tikslas - aptikti kompiuterinio vaizdo briaunas. Tarkime, kad dirbame su vaizdu I. Iš pradžių apskaičiuojami horizontalūs pikselių reikšmių pasikeitimai atliekant pirmosios Sobel operatoriaus matricos konvoliuciją su paveikslėliu I (3.20 formulė). Tuomet apskaičiuojamas vertikalių pikselių reikšmių pasikeitimai atliekant antrosios Sobel operatoriaus matricos konvoliuciją su paveikslėliu I (3.21 formulė). Taip išsiaiškinama, koks yra horizontalus ir vertikalus gradientas, tolimesnis žingsnis – vertikalaus ir horizontalaus gradiento sumos išsiaiškinimas. Kadangi matricose yra neigiamų reikšmių, todėl kai kurių pikselių tam tikros krypties gradientas gali būti neigiamas. Dėl šios priežasties paprasta suma netenka savo prasmės. Problema išsprendžiama pritaikant Pitagoro teoremą (3.22 formulė). Tam tikros Sobel operatorių naudojančios aplinkos formulę 3.22 pakeičia į 3.23.

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * I \tag{3.20}$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * I \tag{3.21}$$

$$G = \sqrt{G_x^2 + G_y^2} \tag{3.22}$$

$$G = |G_x| + |G_y| \tag{3.23}$$

Python programavimo kalbos kompiuterinės vizijos biblioteka OpenCV vietoj Sobel'io operatoriaus matricų G_x, G_y naudoja Scharr'o matricas (atitinkamai formulės 3.24 ir 3.25). Sobel operatorius naudojamas su nespalvotais vaizdais.

$$G_x = \begin{bmatrix} -3 & 0 & 3 \\ -10 & 0 & 10 \\ -3 & 0 & 3 \end{bmatrix} \tag{3.24}$$

$$G_y = \begin{bmatrix} -3 & -10 & -3 \\ 0 & 0 & 0 \\ 3 & 10 & 3 \end{bmatrix} \tag{3.25}$$

Yra galimybė suskaičiuoti gradiento kryptį pasinaudojant 3.26 formule [50].

$$\Theta = \arctan \frac{G_y}{G_x} \tag{3.26}$$

3.9. Canny kraštų aptikimo metodas

Algoritmas dirba su nespalvotais vaizdais, todėl prieš naudojant algoritmą vaizdą reikia pasiversti iš spalvoto į nespalvotą. Canny kraštų aptikimo algoritmas susideda iš penkių žingsnių [15]:

1. Triukšmo pašalinimas;

   Triukšmo šalinimui naudojamas Gauso glodinimo filtras. Gauso glodinimo filtro $(2k+1) \times (2k+1)$ dydžio matricos reikšmių radimo formulė:

   $$H_{ij} = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(i-(k+1))^2 + (j-(k+1))^2}{2\pi\sigma^2}\right); \quad 1 \leq i, j \leq (2k+1) \tag{3.27}$$

2. Gradiento skaičiavimas;

   Gradientai ir jų kryptys yra suskaičiuojami pasinaudojant Sobel operatoriumi. Skaičiuojant naudojamos 3.20 3.21 3.22 3.26 formulės.

3. Nemaksimalios reikšmės suspaudimas;

   Po Sobel operatoriaus panaudojimo kai kurios briaunos vaizde yra storos, o kai kurios plonos. Nemaksimalios reikšmės suspaudimas padeda suploninti storąsias. Algoritmas eina per gradientų intensyvumo matricą ir suranda pikselius su maksimalia reikšme vienoje briaunos kryptyje. Jeigu randamas kaimyninis vienos krypties pikselis su didesne intensyvumo reikšme, tuomet einamojo pikselio intensyvumo reikšmė pakeičiama į 0 (jis tampa juodu). Jeigu nėra kaimyninio vienos krypties pikselio su didesne intensyvumo reikšmė, tuomet iteruojamo pikselio reikšmė paliekama kokia ir buvus. Gautas rezultatas yra paveiksliukas su plonesnėm briaunom. Po šio žingsnio kai kurie pikseliai yra ryškesni nei kiti. Šios savybės pašalinimui reikalingi sekantys du Canny kraštų aptikimo žingsniai.

4. Dvigubo slenksčio panaudojimas;

   Dvigubo slenksčio panaudojimas siekia pikselius išskirti į tris kategorijas: stiprius, silpnus ir neaktualius. Pikseliai, kurių intensyvumo reikšmė yra didesnė už viršutinio slenksčio reikšmę kategorizuojami kaip stiprūs. Pikseliai, kurių intensyvumo reikšmė yra mažesnė už apatinio slenksčio reikšmę kategorizuojami kaip neaktualūs. Visi likę pikseliai, t.y. pikseliai, kurių intensyvumo reikšmės yra tarp apatinio slenksčio ir viršutinio slenksčio ribų pažymimi kaip silpni. Histerezės mechanizmas (sekantis Canny algoritmo žingsnis) identifikuos, kurie iš silpnų pikselių gali būti laikomi stipriais, o kurie neaktualiais. Po šio žingsnio lieka tik dvi pikselių intensyvumo reikšmės - stipriųjų ir silpnųjų.

5. Briaunų aptikimas naudojantis histereze.

   Histerezės procesas perkategorizuoja stiprius pikselius į silpnus. Šis veiksmas įvyksta tada ir tik tada kai aplink iteruojamą pikselį kaimynystėje yra bent vienas stiprus pikselis.

Kai kur literatūroje ketvirtasis ir penktasis žingsniai yra sujungiami į vieną [19].

4. Metodai, taikomi po vaizdo apdorojimo

4.1. Hough apskritimų transformacija

Hough apskritimų transformacija ir jos modifikacijos yra dažniausiai naudojamas apskritimų aptikimo metodas akademinėje bendruomenėje. Hough apskritimų transformacijos algoritmas yra specifinis Hough transformacijos algoritmo atvejis, skirtas aptikti ne tiesėm, o apskritimams. Hough apskritimų transformacija gali būti apibūdinta kaip transformacija iš X,Y-plokštumos į parametrų erdvę [49]. Matematinė apskritimo formulė X,Y-plokštumoje yra apibrėžiama šiuo būdu:

$$r^2 = (x - a)^2 - (y - b)^2 \tag{4.1}$$

Čia r yra apskritimo spindulys, o a ir b yra apskritimo centras. Lygtys 4.2 ir 4.3 yra parametrinės apskritimo lygties reprezentacijos.

$$x = a + r\cos\Theta \tag{4.2}$$

$$y = b + r\sin\Theta \tag{4.3}$$

Apskritimas turi tris parametrus r, a ir b, taigi parametrų erdvė priklauso R^3. Algoritmo supaprastinimo tikslais apskritimo spindulys gali būti laikomas kaip konstanta arba apribotas tam tikrame intervale.

Iš pradžių algoritmas suranda skaitmeninio vaizdo briaunas. Šis žingsnis nesusijęs su Hough'o transformacija. Briaunų suradimui dažnai naudojamos Canny kraštų aptikimo metodas, bet galimas ir Sobel algoritmo ar kitų morfologinių operacijų naudojimas šiam tikslui. Tuomet kiekvienam briaunos taškui yra piešiamas pageidautino spindulio apskritimas. Apskritimas brėžiamas parametrų erdvėje taip, kad x ašies reikšmė lygi a, y ašies reikšmė lygi b, o z ašies reikšmė lygi r. Koordinates, kurios priklauso nubrėžtam apskritimui padidiname vienetu sumatoriaus matricoje (angl. accumulator matrix). Sumatoriaus matrica yra tokio pačio dydžio kaip parametrų erdvė. Taip apeinamas kiekvienas briaunos taškas piešiant apskritimus su pageidautinu spinduliu ir didinant vienetu sumatoriaus reikšmes. Tuomet sumatoriaus matrica turės reikšmes, kurios nusakys, kiek apskritimų eina per individualias koordinates. Taigi aukščiausios matricos reikšmės (viršijančios tam tikrą slenkstinę reikšmę) sutaps su skaitmeniniame vaizde esančiais apskritimų centrais [30].

1 algoritmas. Hough apskritimų transformacijos algoritmas.

1: Surandamos skaitmeninio vaizdo briaunos {Pritaikant Canny kraštų aptikimo ar kitą metodą}
2: {Hough transformacijos algoritmo pradžia.}
3: Kiekvienam briaunos taškui piešiamas apskritimas. Piešiamo apskritimo centras yra einamasis briaunos taškas, o spindulys lygus r. Sumatoriaus matricoje didinamos su nupiešto apskritimu besikertančios koordinatės vienetu.
4: Sumatoriuje parenkamos viena ar kelios maksimalios reikšmės, kurios bus aptiktų apskritimų centrai.
5: {Hough transformacijos algoritmo pabaiga.}
6: Surasti parametrai (r,a,b) sutampantys su parinktom maksimaliom reikšmėm yra pažymimi originaliame skaitmeniniame vaizde.

4.2. Suzuki algoritmas

Kontūrai yra kreivės, jungiančios visus tęstinius taškus, kurie turi tam tikrą savybę, pavyzdžiui taško intensyvumą. Kontūrų išgavimas gali būti tarpinis žingsnis formų analizės, įvairių objektų aptikimo ir atpažinimo uždaviniuose. Yra įvairių kontūrų aptikimo algoritmų kaip kad Theo Pavlidis ar Kovalevsky algoritmai, bet šiame skyriuje bus nagrinėjamas Suzuki algoritmas, kurį naudoja Python programavimo kalbos OpenCV ir kitos vaizdų apdorojimo bibliotekos.

Suzuki algoritmas buvo vienas pirmųjų algoritmų, kuris apibrėžė hierarchinį santykį tarp kontūrų. Šis algoritmas taip pat diferencijuoja kontūrus į išorinius ir vidinius. Tarkime, kad $f_{i,j}$ žymi pikselio reikšmę su koordinatėmis (i, j). Viršutinė ir apatinės eilės bei kairiausias ir dešiniausias stulpelis sudaro paveikslėlio rėmą. Šiuo būdu Suzuki algoritmas priskiria unikalų skaičių, pažymėtą NBD, kiekvienam naujam rastam kontūrui. Paveikslėlio rėmo NBD reikšmė lygi 1. Taip pat saugoma informacija apie tėvinį, t.y. paskutinį rastą kontūrą, pažymėtą LNBD.

Suzukio algoritmo aprašymas remiasi šiuo straipsniu [41]. Straipsnyje taip pat pateikiamas algoritmas gauti vien išorinį kontūrą.

2 algoritmas. Suzuki algoritmas.

{Pirmasis algoritmo žingsnis. Kontūro nustatymas.}
(1.) Atliekamas sąlyginis sakinys:
if $f_{ij} = 1$ ir $f_{i,j-1} = 0$ (t.y. jeigu einamasis pikselis (i, j) yra išorinis kontūras) **then**
NBD reikšmė padidinama vienetu ir $(i_2, j_2) \leftarrow (i, j-1)$
else if $f_{ij} \geq 1$ ir $f_{i,j+1} = 0$ (t.y. jeigu einamasis pikselis (i, j) yra vidinis kontūras) **then**
NBD reikšmė padidinama vienetu ir $(i_2, j_2) \leftarrow (i, j+1)$, ir jeigu $f_{i,j}$, tuomet LNBD $\leftarrow f_{i,j}$
else
Keliaujama į (3).
end if
{Antrasis algoritmo žingsnis. Kontūro sekimas nuo pradžios taško (i, j).}
(2.1) Pradedant nuo i_2, j_2 apžvelgiami kaimyniniai (i, j) pikseliai pagal laikrodžio rodyklę, kol randamas pirmas nenulinės reikšmės pikselis. Tarkim (i_1, j_1) yra pirmas rastas nenulinis pikselis. Jeigu nerasta nenulinių pikselų, priskiriama -NBD reikšmė $f_{i,j}$ ir keliaujama į (3).
(2.2 $(i_2, j_2) \leftarrow (i_1, j_1)$ ir $(i_3, j_3) \leftarrow (i, j)$)
(2.3) Pradedant nuo kito (i_2, j_2) pikselio elemento (i_3, j_3) kaiminystėje prieš laikrodžio rodyklę randamas pirmas nenulinis pikselis (i_4, j_4).
(2.4) Atliekamas sąlyginis sakinys, kuris pakeičia (i_3, j_3) pikselio reikšmę f_{i_3,j_3}:
if $i_3, j_3 + 1$ yra 0 reikšmės pikselis aptiktas (2.3) **then**
$f_{i_3,j_3} \leftarrow -NBD$.
else if $i_3, j_3 + 1$ nėra 0 reikšmės pikselis aptiktas (2.3) **then**
$f_{i_3,j_3} \leftarrow NBD$.
else
f_{i_3,j_3} reikšmė nėra keičiama.
end if
(2.5) Jeigu $(i_4, j_4) = (i, j)$ ir $(i_3, j_3) = (i_1, j_1)$, tuomet keliaujama (3); kitu atveju $(i_2, j_2) \leftarrow (i_3, j_3)$, $(i_3, j_3) \leftarrow (i_4, j_4)$ ir keliaujama (3.3)
{Trečiasis algoritmo žingsnis. Iteravimas per vaizdą.}
(3.1.) Atliekamas sąlyginis sakinys:
if $f_{i,j} \neq 1$ **then**
LNBD $\leftarrow |f_{i,j}|$
end if
(3.2.) Skenuojamas tolimesnis pikselis $(i, j + 1)$. Algoritmas baigia darbą, kuomet pasiekiamas dešiniausias apatinysis vaizdo pikselis.

4.3. Konvoliucinis neuroninis tinklas

Konvoliucinis neuroninis tinklas (angl. convolutional neural network, trump. CNN) yra neuroninių tinklų tipas. kuris skirtas kompiuterinės regos uždaviniam spręsti kaip kad klasifikavimas, segmentavimas, objektų aptikimas. Neuroniniai tinklai yra algoritmai naudojami dėsningumų duomenyse atpažinimui. Neuroniniai tinklai sudaryti iš neuronų (funkcijos turinčios daug įvesties parametrų ir viena išvestį), kurie sudaro atskirus sluoksnius (neuronų kolekcija). Skirtumas tarp neuroninio tinklo ir KNT yra konvoliucinis sluoksnis, kuris atlieka konvoliucijos operaciją (žr. 3.4 skyrių). Techniškai konvoliucijos operacija konvoliucinių neuroninių tinklų kontekste yra kryžminės koreliacijos operacija, kuri tik konvoliucninių neuroninių tinklų kontekste vadinama konvo-

liucija. Daug mašininio mokymosi bibliotekų konvoliucijos operaciją įgyvendina kaip kryžminę koreliaciją [23].

Įprastai KNT kartu su konvoliuciniu sluoksniu eina aktyvacijos sluoksnis (angl. activation layer) ir sutelkimo sluoksnis (angl. pooling layer). Konvoliucijos sluoksnis išgauna požymių schemą, aktyvacijos sluoksnis pašalina tiesiškumą modelyje pritaikydamas aktyvacijos funkciją požymių schemos matricai, o sutelkimo sluoksnis sumažina įvesties matricos dydį. Konvoliucijos filtras sistematiškai pritaikomas kiekvienai atitinkamų dimensijų skaitmeninio vaizdo daliai taip vaizde atpažįstant tam tikrus požymius ir gaunant požymių schemą (angl. feature map). Turint požymių schemą modelyje pašalinamas tiesiškumas aktyvacijos sluoksnyje pritaikius aktyvacijos funkciją pavyzdžiui ReLU (angl. Rectified Linear Unit). Konvoliucijos operacija buvo naudojama dar prieš konvoliucinius neuroninius tinklus, kad būtų surasti tam tikri požymiai skaitmeniniame vaizde (žr. 3.8 skyrių), tačiau konvoliucijos operacijos pritaikymas KNT skiriasi tuo, kad filtro svoriai yra išmokstami apmokant neuroninį tinklą, t.y. jie iš anksto nežinomi.

5. Objektų aptikimo metodų įvertinimo metrikos

5.1. Painiavos matrica

Painiavos matrica yra technika įvertinti, koks geras yra klasifikavimo modelis. Painiavos matrica apibendrina klasifikavimo problemos prognozavimo rezultatus matricos pavidalu. Teisingų ir neteisingų prognozių kiekiai yra išskaidomi į atskiras klases ir sudedami. Painiavos matrica parodo kaip klasifikavimo modelis darydamas prognozes susipainioja. Ji padeda atpažinti, kokias klaidas prognozėje daro modelis. Norint gauti painiavos matricą yra atliekamos klasifikavimo modelio prognozės ir gautos reikšmės yra surašomos į šią lentelę [26]:

1 lentelė. Painiavos matrica.

		Tikrosios reikšmės	
		Teigiamos	Neigiamos
Prognozuojamos reikšmės	Teigiamos	TP	FP
	Neigiamos	FN	TN

Matricoje esančių keturių kintamųjų paaiškinimas:

- TP (angl. True Positive) – atvejų skaičius, kuomet teigiamos reikšmės prognozuojamos kaip teigiamos.
- TN (angl. True Negative) – atvejų skaičius, kuomet neigiamos reikšmės prognozuojamos kaip neigiamos.
- FP (angl. False Positive) – atvejų skaičius, kuomet neigiamos reikšmės prognozuojamos kaip teigiamos. Dar vadinama pirmo tipo klaida (teisingos nulinės hipotezės atmetimas).
- FN (angl. False Negative) – atvejų skaičius, kuomet teigiamos reikšmės prognozuojamos kaip neigiamos. Dar vadinama antro tipo klaida (neteisingos nulinės hipotezės neatmetimas).

Painiavos matricos kintamieji leidžia apibrėžti klasifikavimo modelio įvertinimą atliekančias metrikas:

- $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$, metrika apibrėžia visų teisingų prognozių santykį su visomis prognozėmis.
- $Precision = \frac{TP}{TP+FP}$, metrika apibrėžia visų teisingų teigiamų prognozių santykį su visomis teigiamomis prognozėmis.
- $Recall = \frac{TP}{TP+FN}$, metrika apibrėžia tikimybę, kad teigiama prognozė bus teisinga.
- $F_1 = 2 * \frac{precision*recall}{precision+recall}$ (dar vadinama F-score), metrika yra harmoninis $Precision$ ir $Recall$ vidurkis.

5.2. mAP, AR, metrikų apibrėžimų skirtumai

Vidutinių tikslumų vidurkis (angl. Mean Average Precision, trump. mAP) yra metrika, kuri naudojama objektų aptikimo ir vaizdų segmentavimo problemų sprendimo analizei. Daug objektų

aptikimo modelių naudoja mAP metriką modelių įvertinime, pavyzdžiui, Faster R-CNN, MobileNet SSD ir YOLO. Vidutinių tikslumų vidurkio formulės skaičiavimui reikalingos keturios kitos submetrikos: painiavos matrica, Recall, Precision (žr. 5.1 skyrių), sankirtos ir sąjungos santykis (angl. Intersection over Union, trump. IoU). Tarkime, kad A yra aptikto objekto spėjimo ribojantis stačiakampis (angl. bounding box), o B yra teisingas (angl. ground truth) ribojantis stačiakampis, tuomet sankirtos ir sąjungos santykis apibrėžiamas šia formule:

$$IoU = \frac{|A \cap B|}{|A \cup B|} = \frac{|I|}{|U|} \tag{5.1}$$

Painiavos matricoje esančios reikšmės nustatomos atsižvelgiant į tai ar IoU reikšmes viršija tam tikrą slenkstinę reikšmę. Mažesnės IoU reikšmės reiškia aukštesnes $Recall$ reikšmes ir žemesnes $Precision$ reikšmes, o aukštenės IoU reikšmės reiškia žemesnes $Recall$ reikšmes ir aukštesnes $Precision$ reikšmes. $Precision$ apibrėžia kokia dalis iš visų modelio ribojančių stačiakampio pranašysčių yra taisyklinga. $Recall$ apibrėžia kokia dalis iš visų tikrų ribojančių stačiakampių modelio buvo išpranašauta taisyklingai. Priklausomai nuo modelio naudojimo tikslo skirtingi modeliai vertina skirtingas metrikas. Pavyzdžiui dirbtinio intelekto valdomoms mašinoms svarbesnė $Recall$ metrika, nes svarbiau taisyklingai aptikti didesnę dalį žmonių. Modeliui, kuris klientui naudojamoje platformoje rekomenduoja filmus, muziką ar klipus svarbiau, kad iš visų pasiūlymų kuo didesnė dalis būtų taisyklinga, todėl labiau vertinama $Precision$ metrika. TN vienos klasės objektų aptikimo užduoties kontekste neegzistuoja, nes tai neturi prasmės. Tai reikštų, kad modelis taisyklingai nenumato neegzistuojančio ribojančio stačiakampio, ši metrika nebus skaičiuojama.

Vidutinių tikslumų vidurkis nėra tikslumų vidurkis kaip kad būtų galima suprasti iš pavadinimo, supaprastinant galima sakyti, kad tai yra plotų po $precision - recall$ kreivėmis vidurkis. Norint suskaičiuoti mAP reikia suskaičiuoti tikslumų vidurkį (angl. Average Precisio, trump. AP) kiekvienai klasei atskirai. Nagrinėjamos klasės modelio ribojančių stačiakampių pranašystės yra surikiuojamos mažėjimo tvarka pagal pranašystės tikimybes į lentelė ir paeiliui yra suskaičiuojamos $Precision$ ir $Recall$ reikšmės. Tuomet gautos $Precision$ ir $Recall$ reikšmės yra sužymimoms grafike taip gaunant $precision-recall$ kreivę. Klasės AP yra plotas po nagrinėjamos klasės $precision - recall$ kreive. Tarkime, kad AP_i yra objekto klasės vidutinis tikslumas, tuomet mAP gali būti apibrėžiamas šia formule:

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{5.2}$$

Nagrinėjant mAP metriką svarbu suprasti kokiame kontekste aptariamos jos reikšmės, nes jos gali būti skaičiuojamos skirtingai, pakeičiant AP ir mAP skaičiavimo apibrėžimos. Tikslios modelių metrikos yra apibrėžiamos pagal objektų aptikimo užduoties viešus iššūkius. PASCAL Visual Object Classes Challenge 2005 naudojo ROC-AUC metriką, PASCAL Visual Object Classes Challenge 2007 pristatė 11 taškų interpoliacijos metodą AP skaičiavime, PASCAL Visual Object Classes Challenge 2010 [17] pristatė PR-AUC, kuomet skaičiuojant AP yra atsižvelgiama į visą plotą (2011 ir 2012m. mAP reikšmės nuo 2010m. nesiskiria), t. y. $Precision$ reikšmės nebėra interpoliuojamos, MS COCO 2014 [16] interpeliuoja $precision - recall$ reikšmę 101 taške ir apskaičiuoja mAP reikšmę su 10 skirtingų IoU slenksčių nuo 0,5 iki 0,95 vis didinant slenksčio reikšmę 0,05. Toks MS COCO 2014 mAP apibrėžimas literatūroje dar gali būti užrašomas kaip

mAP@[0.5:.05:.95]. Šiuo metu COCO mAP yra populiariausia objektų aptikimo modelių įvertinimo metrika. Nepriklausomai nuo mAP apibrėžimo visos reikšmės pakliūna į intervalą nuo 0 iki 1 imtinai, todėl kai kur literatūroje mAP išreiškiama procentalia išraiška.

MS COCO 2014 be mAP dar apibrėžia kitas 11 metrikų: mAP@.50, mAP@.75, mAP (small), mAP (medium), mAP (large), AR@1, AR@10, AR@100, AR@100 (small), AR@100 (medium), AR@100 (large). mAP skaičiavimas dar atliekamas mažiem (plotas mažesnis už 32*32 pikselius), vidutiniam (plotas didesnis už 32*32 pikselius ir mažesnis už 96*96 pikselius) ir dideliem (plotas didesnis už 96*96 pikselius) ribojantiems stačiakampiams. AR (angl. average recall) metrika yra suvidurkinta didžiausių $Recall$ reikšmė fiksuotam skaičiui aptikimų vaizde per skirtingas kategorijas ir IoU slenksčius. Fiksuotas skaičius aptikimų rašomas po eta simbolio. Pavyzdžiui, AR@1 reiškia suvidurkintą didžiausių $Recall$ reikšmę aptinkant vieną objektą vaizde ir pereinant per skirtingas objektų kategorijas bei IoU slenksčius nuo 0,5 iki 0,95 su žingsniu 0,05.

6. Defektų atpažinimas

6.1. Tyrimo duomenys

Visi AJM membranų paviršiaus su defektais vaizdų duomenys gauti iš Vilniaus universiteto Gyvybės mokslų centro mokslo darbuotojos Marijos Jankunec. Tyrimo duomenys pateikti kartu su darbu MIF Drive darbo autoriui skirtoje direktorijoje. Įkeltoje elektroninėje darbo versijoje src/ direktorijoje esančiose direktorijose labeled/ ir unlabeled/ yra visi tyrime naudoti duomenys. Direktorijose yra atitinkamai dalis Gyvybės mokslų centro mokslo darbuotojų sužymėtų ir nesužymėtų vaizdų. Vaizdų sužymėjimas atliktas Pascal VOC XML formatu. Duomenys sudėti į atskiras direktorijas pagal vaizdo gavimo datą, nufotografuoto ląstelės plotą μ (μ yra SI sistemos priešdėlis mikro, kuris reiškia daugiklį 10^{-6}) ir vaizdo rezoliuciją. Pavyzdžiui "20190114_1um_1024px" reiškia, kad nuotrauka daryta 2019-01-14, nufotografuotas ląstelės plotas yra 1 μm x 1 μm ir vaizdo raiška yra lygi 1024 x 1024px (žr. 2 lentelę). Direktorijose pateikti specialaus RAW formato, kuris grąžinamas iš AJM, vaizdai, skirtingo kanalo vaizdai (aukščio, fazės, amplitudės), sužymėtos koordinatės (Pascal VOC XML formatu ir CSV failais su defektų centro koordinatėmis), skriptai RAW failų apdorojimui (žr. 3 lentelę). Kadangi aukščio kanalo vaizdų lyginant su kitais kanalais buvo daugiausia tyrime pasirinkta aukščio kanalo vaizdų analizė.

2 lentelė. Vaizdų charakteristikos.

Data	Plotas	Rezoliucija	RAW vaizdų kiekis	Aukščio kanalo vaizdų kiekis
2019-01-14	1 μm x 1 μm	1024 x 1024px	11	9
2019-01-14	2 μm x 2 μm	512 x 512px	1	1
2019-01-14	2 μm x 2 μm	1024 x 1024px	1	1
2019-03-27	5 μm x 5 μm	512 x 512px	1	1
2019-04-02	5 μm x 5 μm	512 x 512px	7	3
2019-07-30	2 μm x 2 μm	512 x 512px	9	9
2019-08-28	2 μm x 2 μm	512 x 512px	13	9
2019-08-28	2 μm x 2 μm	512 x 512px	11	9
2019-09-12	2 μm x 2 μm	512 x 512px	0	9
2018-02-04	1 μm x 1 μm	1024 x 1024px	9	9
2020-11-26	1 μm x 1 μm	576 x 576px	0	30

Vaizdų plotas varijuoja nuo 1 μm x 1 μm iki 5 μm x 5 μm. Vaizdai yra trijų rezoliucijų: 512 x 512px, 576 x 576px, 1024 x 1024px. Vaizdai surinkti per 2018-2020 metų laikotarpį. Vaizdai yra trijų kanalų (aukščio, fazės, amplitudės), dalis gautų aukščio kanalo vaizdų yra konvertuoti į spalvotus pasinaudojus $scikit - image(skimage)$ atviro kodo $Python$ programavimo kalbos biblioteka. Sužymėtų vaizdų direktorijoje yra 57 RAW formato vaizdai, 51 aukščio kanalo PNG formato skaitmeninis vaizdas. Yra 18 amplitudės kanalo vaizdų, 9 iš jų PNG formato, 9 iš jų TIF formato. Taip pat yra 18 fazės kanalo vaizdų, 9 iš jų PNG formato, 9 iš jų TIF formato. 5 vaizdai buvo sužymėti netaisyklingai. Nesužymėtų vaizdų direktorijoje yra 25 RAW formato vaizdai, 53 aukščio kanalo skaitmeniniai vaizdai, 1 fazės kanalo vaizdas, 3 identiški vaizdai be meta duomenų, kurie apdirbti skirtingais išankstinio vaizdų apdorojimo metodais. 34 vaizdai yra TIF formato, o

3 lentelė. Tyrimo duomenų direktorijų aprašymas.

Pavadinimas	Tipas	Kiekvienoje direktorijoje	Aprašymas
.ipynb_checkpoints	Direktorija	Taip	Pateikiamas ipynb formato Python kodas, kuriuo iš _raw direktorijoje esančių tif formato vaizdų bei tekstinių failų sukuriama height direktorija su png formato vaizdais ir pascal_voc direktorija su šio formato žymėjimais.
_raw	Direktorija	Taip	RAW formato failai, kurie grąžinami iš atominės jėgos mikroskopo. Taip pat iš RAW formato eksportuoti TIF formato vaizdai su sužymėtomis koordinatėmis.
amplitude	Direktorija	Ne	Amplitudės kanalo vaizdai PNG formatu.
coords	Direktorija	Ne	CSV formato failai su sužymėtų defektų centro koordinatėmis.
height	Direktorija	Taip	Aukščio kanalo vaizdai PNG formatu.
height_false_color	Direktorija	Ne	Spalvoti aukščio kanalo vaizdai PNG formatu.
pascal_voc	Direktorija	Taip	Pascal VOC XML formatu sužymėti defektai.
phase	Direktorija	Ne	Fazės kanalo vaizdai PNG formatu.
fragment_map.txt	Failas	Ne	Tekstinis failas, kuriame nurodyta, kaip turi būti sujungti direktorijoje esantys vaizdai į vieną vaizdą.
parse.ipynb	Failas	Taip	ipynb formato Python kodas, kuriuo pasinaudojus skimage biblioteka nuspalvinami height/ direktorijoje esantys aukščio kanalo vaizdai. Dalyje direktorijų failas dubliuojasi su .ipynb_checkpoints direktorijoje esančiu kodu.

likusieji 23 yra PNG formato. Tolimesniam tyrimo darbui TIF formato failai buvo konvertuoti į PNG formatą su $ImageMagick$ atviro kodo programine įranga. Dvejuose failuose buvo sužymėti defektai klasterio viduje.

Atmetus vaizdus, kurių darbo autorius negalėjo sužymėti dėl per didelio kiekio defektų iš viso tyrime naudojama 90 vaizdų. 46 iš jų jau buvo sužymėti. Likusieji 44 sužymėti darbo autorius naudojantis $LabelImg$ atviro kodo programine įranga Pascal VOC XML formatu. Tyrime dirbta su Pascal VOC XML, YOLO Darknet TXT, Tensorflow Object Detection CSV ir Tensorflow TFRecord skaitmeninių vaizdų žymėjimo formatais. Sužymėti Pascal VOC XML formato duomenys su kuriais dirbama tyrime buvo konvertuojami į YOLO Darknet TXT formatą pasinaudojus $Roboflow$ kompanijos programinės įrangos teikiamomis paslaugomis. Tyrime naudoti vaizdai vidutiniškai turėjo 81,82 defekto vienoje AJM nuotraukoje. Mažiausiai vaizde buvo 6 defektai, o daugiausiai 908. Iš viso buvo sužymėti 7364 defektai.

6.2. Programavimo darbų aplinka

Tyrimo darbas atliktas naudojantis $Python$ programavimo kalba (3.9.7 versija). $Python$ programavimo kalbos bibliotekos, kuriomis buvo naudotasi tyrimo metu:

- $OpenCV$ [6]. Biblioteka yra skirta realaus laiko kompiuterinės regos užduotims spręsti. Su šia biblioteka darbe atlikti išankstiniai vaizdų apdorojimo metodai, defektų vizualizavimas, defektų aptikimas.

- $TensorFlow$ [10] Biblioteka yra atviro kodo ir ji skirta mašininio mokymosi bei dirbtinio intelekto užduotims spręsti. Su šia biblioteka buvo apmokyti tyrime nagrinėjami modeliai (išskyrus YOLOv4) ir atliktas modelių statistikų vizualizavimas tolimesnei analizei su $TensorBoard$ įrankiu, kuris yra $TensorFlow$ bibliotekos dalis.

- $NumPy$ [5]. Biblioteka yra skirta greitesniam darbui su masyvais nei standartiniai $Python$ masyvai, ji naudojama moksliniuose skaičiavimuose. Su šia biblioteka buvo apdorojami masyvuose esantys duomenys, pavyzdžiui, atliekant Hough apskritimų transformaciją ar YOLOv4 modelio defektų aptikimą.

- $imgaug$ [25]. Biblioteka yra skirta skaitmeninių vaizdų duomenų padidinimui mašininio mokymosi eksperimentuose. Su šia biblioteka sukurtas įrankis atlikti duomenų plėtimo operacijas.

- $Matplotlib$ [4]. Biblioteka yra skirta duomenų vizualizavimui. Su šia biblioteka buvo vizualizuotos defektų centrų koordinatės.

- $Pandas$ [7]. Biblioteka yra skirta skirta duomenų apdorojimui ir analizei. Su šia biblioteka buvo keičiamos duomenų struktūros ir nuskaitomi duomenys atliekant objektų sužymėjimo formatų keitimą.

- PIL [8] (angl. **P**ython **I**maging **L**ibrary). Biblioteka yra skirta skirtingų vaizdo failų formatų atidarymui, apdorojimui ir išsaugojimui. Su šia biblioteka atliktos vaizdinio failo atidarimo ir dimensijų gavimo operacijos.

- Kitos: os, io, sys, csv, argparse, glob, logging.

Tyrimo metu buvo naudojamasi Microsoft Windows 11 Pro operacine sistema. Sistemos tipas yra 64 bitų. Dirbant su šia operacine sistema naudotasi $Anaconda$ (2021.11) programine įranga, kuri supaprastina paketų valdymą. Naudotasi $Conda$ (4.10.3 versija) paketų valdytoju, kuris yra $Anaconda$ viduje. Tam tikrais atvejais naudotasi darbalaukio reikmėms pritaikyta Linux šeimos operacine sistema $Ubuntu$ (22.04.1 versija) sukuriant virtualią mašiną su Oracle VM VirtualBox (6.1.32 versija). $Ubuntu$ sistemos tipas yra 64 bitų.

Modelių apmokymo skaičiavimai atlikti Google Colab debesų kompiuterijos aplinkoje. Google Colab aplinkos pasirinkimą lėmė grafinio procesoriaus (angl. graphics processing unit, trump. GPU) neturėjimas. Modelių apmokymų skaičiavimai atlikti su Tesla T4 grafiniu procesorium. Darbo autorius tyrimo darbui įsigijo Colab Pro aplinkos prieigą dėl nepertraukiamų skaičiavimų patogumo. Visi tyrimo skaičiavimai gali būti pakartojami nemokamoje Google Colab debesų kompiuterijos aplinkoje. Naudojamo kodo komentaruose parašytos komandos su instrukcijomis, kuriomis reikėtų pratęsti modelių apmokymų skaičiavimus išnaudotos kvotos atvejais. Vaizdų žymėjimas atliktas naudojantis $LabelImg$ [33] (1.8.6 versija) atviro kodo programine įranga. Duomenų plėtimas atliktas naudojantis Roboflow [9] programine įranga. YOLOv4 modelis apmokytas naudojantis Darknet karkasu [32]. Modelio metrikos gautos su Object Detection Metrics GitHub repozitorija [34].

6.3. Programavimo darbai

Tyrimo metu su Python programavimo kalba sukurti programavimo įrankiai:

- def_custom. Įrankis yra ipynb formato kodas, kuris nuskaito Google Drive patalpintą custom_def_model/ direktoriją su paruoštus duomenis talpinančia ir suarchyvuota Darknet [32] repozitorija. Kodas sukompiliuoja Darknet karkasą ir atlieka modelio apmokymą.
- $custom\_def\_yolov4$. Skriptas, kuris nuskaito YOLOv4 modelio svorių reikšmes ir konfigūraciją bei atlieka objektų aptikimą vaizde.
- get_det_res. Skriptas, kuris paruošia aptinkamų defektų ribojančius stačiakampius reikiamu formatu, kad objektų aptikimo metrikų repozitorija [34] galėtų atlikti skaičiavimus.
- $tf2\_custom$. Įrankis yra ipynb formato kodas, kuris nuskaito Google Drive patalpintą tf2/ direktoriją su apmokymui ir testavimui skirtais Tensorflow TFRecord (atitinkamai train.record ir test.record failai) bei atlieka aplinkos paruošimo, modelio parsiuntimo, modelio apmokymo ir statistikų vizualizavimo su $TensorBoard$ darbus.
- $process$. Skriptas, kuris atlieka išankstinius vaizdų apdorojimo metodus ir aptinka defektus Suzuki ir Hough apskritimų transformacijos algoritmais.
- $get\_canny\_otsu\_sobel.py$. Skriptas, kuris atlieka Canny kraštų aptikimo, Otsu ar Sobel metodus duomenų aibei.
- $augment$. Skriptas, kuris atlieka duomenų praplėtimo veiksmą (angl. data augmentation).
- $divide$. Skriptas, kuris išskaido vaizdą į nurodyto eilučių ir stulpelių kiekio stačiakampius.
- $draw\_dots$. Skriptas, kuris nuskaito csv faile sužymėtas defektų centro koordinates ir jas sužymi vaizduose.

Tyrimo metu naudoti kitų autorių programavimo įrankiai:

- $detect\_objects$ iš Github repozitorijos paimtas [35] ir modifikuotas įrankis. Skriptas, kuris nuskaito paruoštą Tensorflow 2 produkciniam naudojimui modelį ir aptinka skaitmeniniame vaizde objektus, juos sužymi.

- $generate\_tfrecords$ iš Github repozitorijos paimtas [37] ir modifikuotas įrankis. Skriptas, kuris nuskaito Tensorflow Object Detection CSV formatą su Label map ir konvertuoja formatą į Tensorflow TFRecord.

- xml_to_csv iš Github repozitorijos paimtas [36] ir modifikuotas įrankis. Skriptas, kuris nuskaito Pascal VOC XML formatą ir konvertuoja jį į Tensorflow Object Detection CSV formatą.

6.4. Duomenų paruošimo darbai

Visi 90 tyrimo metu naudojami vaizdai su Pascal VOC XML formato žymėjimais gali būti apdorojami programavimo darbų skiltyje aprašytais įrankiais (žr. 6.3 skyrių) arba Roboflow programine įranga. Šiame darbe aprašytuose skaičiavimuose duomenų apdorojimo darbai atlikti su Roboflow įranga. Tokį sprendimą nulėmė galimybės su Roboflow konvertuoti Pascal VOC XML formato žymėjimus į YOLO Darknet TXT formato žymėjimus ir duomenų praplėtimo galimybės žymėjimo failuose. Norint pakartoti atliktus skaičiavimus be Roboflow programinės įrangos naudojimo reikėtų formatus konvertuojančio įrankio arba atskiro defektų sužymėjimo YOLO Darknet TXT formatu ir atskiro žymėjimo praplėstų vaizdų aibei.

Tyrimo metu surinkti ir pabaigti žymėti Pascal VOC XML formatu 90 aukščio kanalo vaizdų buvo atskirti į apmokymo ir testavimo aibes. Duomenys atskirti į apmokymo ir testavimo aibes laikantis 80/20 taisyklės, kuri sako, kad 80% duomenų aibės turėtų būti skirta modelio apmokymui, o 20% duomenų turi būti taikomi modelio testavimui, t. y. duomenys, kurių modelis apmokymo metu nemato. Taip į duomenų aibę atsitiktinai buvo parinkti 72 (80%) vaizdai ir į testavimo aibę likę 18 (20%) vaizdų. Dėl mažo kiekio vaizdų kyla gresmė modelio persimokymui (angl. model overfitting). Ši problema išsprendžiama atlikus duomenų plėtimo (angl. data augmentation) operaciją modelio apmokymo duomenų aibei. Iš pradžių visiem vaizdam buvo atlikti išankstinio vaizdų apdorojimo metodai ir tada praplėsta apmokymo duomenų aibė.

Atlikti išankstiniai vaizdų apdorojimo metodai (atlikti ir apmokymo ir testavimo duomenų aibėms):

- Pikselių duomenų auto orientacija (angl. auto-orientation of pixel data). Daugelis kamerų laiko pikselius taip pat nepriklausomai kokiame rėžime (pavyzdžiui, horizontaliai ar vertikaliai) turi būti rodomas vaizdas. Tik pagal atskirus EXIF formatu laikomus metaduomenis kamera supranta kaip rodyti vaizdą. Šis vaizdo apdorojimo metodas pašalina EXIF duomenis ir standartizuoja pikselių tvarką. Pikselių duomenų auto orientacija šiuo konkrečiu atveju nėra būtina ir be jos galima būtų pakartoti tyrimo rezultatus, bet tai yra gera praktika, kuri padeda išvengti potencialių klaidų.

- Vaizdų rezoliucija pakeista į 512 x 512px. Tokia rezoliucija pasirinkta, nes dauguma nagrinėjamų vaizdų yra 512 x 512px rezoliucijos ir mažoji dalis yra 576 x 576px bei 1024 x 1024px rezoliucijų. Giliojo mokymosi modelių apmokymas su mažesnės rezoliucijos vaizdais yra greitesnis.

Modelio apmokymo duomenų aibė buvo išplėsta į tris kartus didesnę. Apmokymo duomenų aibėje po duomenų praplėtimo buvo 216 (72* 3) modelio apmokymui reikalingi vaizdai. Iš kiekvieno modelio apmokymo duomenų aibės vaizdo buvo sukurti trys vaizdai vadovaujantis šiomis vaizdo transformacijos taisyklėmis:

- 50% horizontalaus apvertimo (angl. flip) tikimybė.
- 50% vertikalaus apvertimo tikimybė.
- Lygi tikimybė vienam iš šių 90 laipsnių pasukimui: jokio pasukimo, pasukimas pagal laikrodžio rodyklę, pasukimas prieš laikrodžio rodyklę.
- Vaizdas apkerpamas nuo 0 iki 20 procentų (vaizdas priartinamas).

Po visų atliktų paruošimo darbų duomenų aibė buvo sudaryta iš 234 skaitmeninių vaizdų. 216 (80%) iš jų priklausė apmokymo aibei, o 18 iš jų testavimo aibei (20%). Tyrime vaizdai buvo išsaugoti kartu su defektų sužymėjimais Pascal VOC XML, YOLO Darknet TXT, Tensorflow Object Detection CSV ir Tensorflow TFRecord formatais. Tensorflow TFRecord formatas buvo naudojamas darbui su $Tensorflow$ modeliais, YOLO Darknet TXT buvo panaudotas YOLOv4 modelio apmokymui, Pascal VOC XML formatas buvo panaudotas modelio įvertinimo metrikoms gauti, o Tensorflow Object Detection CSV yra imitacija formato, kuris būtų gaunamas verčiant Pascal VOC XML formatą į Tensorflow TFRecord formatą vadovaujantis prie darbo prisegtais programavimo įrankiais.

6.5. Defektų atpažinimas

Defektų atpažinimo uždavinys šiame tyrime buvo sprendžiamas naudojantis septyniais skirtingais metodais. Naudojamus metodus būtų galima išskirti į dvi grupes pagal tai ar jie naudojasi giliojo mokymosi metodikomis ar ne. Iš septynių atpažinimo metodų du nesinaudoja giliojo mokymosi metodikomis ir penki naudojasi. Tyrime nagrinėjami ne giliojo mokymosi defektų atpažinimo metodai: Hough apskritimų transformacija ir Suzuki algoritmas. Šios grupės metodai įgyvendinti naudojantis OpenCV Python programavimo kalbos biblioteka. Tyrime nagrinėjami giliojo mokymosi defektų atpažinimo metodai: YOLOv4, CenterNet HourGlass104 512x512, EfficientDet D0 512x512, SSD ResNet152 V1 FPN 640x640 (RetinaNet152) ir Faster R-CNN Inception ResNet V2 640x640 modeliai. Modeliai buvo apmokyti atpažinti vienos klasės vaizdus - defektus. Visų tyrimo metodų defektų aptikimo vizualizacijos su dalimi testavimo aibės vaizdų gali būti rastos darbo prieduose (žr. E priedą).

Hough apskritimų transformacijos ir Suzuki algoritmai buvo suprogramuoti taip, kad prieš aptikdami defektus atliktų vaizdo apdorojimo metodus. Vaizdo apdorojimo metodų naudojimas ir metodų parametrų pasirinkimas argumentuotas be metrikų, vien atsižvelgiant į pavienių vaizdų defektų aptikimą. Prieš Hough apskritimų transformacijos skaičiavimus vaizdas buvo apdorotas Otsu segmentavimo metodu, jis nereikalauja parametrų kaip kad Canny briaunų radimo algoritmas ar viršutinis slenkstinis filtravimas. Hough apskritimų transformacijos analizės skaičiavimuose mažiausia spindulio parametro r reikšmė buvo 1, o didžiausia 13, slenksčio analizuojamai briaunai parametras buvo lygus 50, o slenksčio apskritimo aptikimui sumatoriaus matricoje reikšmė nustatyta į 20, atstumas tarp spindulių ne mažesnis už 10. Prieš Suzuki algoritmą buvo atliktas viršutinis slenkstinis filtravimas su T parametro reikšme 127, kad toliau būtų dirbama su binariniu vaizdu.

YOLOv4 modelio apmokymas buvo nustatytas 2000 iteracijų ir truko 5 valandas 35 minutes. CenterNet HourGlass104 512x512, EfficientDet D0 512x512, SSD ResNet152 V1 FPN 640x640 (RetinaNet152) ir Faster R-CNN Inception ResNet V2 640x640 modeliai buvo apmokyti per 8000 iteracijų. CenterNet HourGlass104 512x512 modelio apmokymas užtruko 3 valandas 35 minutes, EfficientDet D0 512x512 35 minutes, SSD ResNet152 V1 FPN 640x640 (RetinaNet152) 1 valandą 41 minutę ir Faster R-CNN Inception ResNet V2 640x640 modelis 2 valandas 49 minutes. YOLO v4 partijos (angl. batch) reikšmė buvo nustatyta į 64 (žr. lentelę 4), Faster R-CNN Inception ResNet V2 640x640 į 2, o kituose modeliuose į 4. Pasirinkimą nulėmė modelio metrikų pagerėjimas eksperimentuojant su skirtingomis reikšmėmis ir techninės galimybės.

4 lentelė. YOLOv4 modelio konfigūracijos nustatymų pagal nutylėjimą pakeitimai.

Konfigūracijos parametras	Priskirta parametro reikšmė
batch	64
subdivisions	64
width	416
height	416
max_batch	2000
steps	1800
filters	18
classes	1

Pasinaudojus objektų aptikimo uždavinio modelių įvertinimo įrankiu buvo apskaičiuotos PASCAL VOC 2012 ir MS COCO vertinimo metrikos. Kadangi Hough apskritimų transformacija ir Suzuki algoritmas nepateikia aptinkamų defektų atpažinimo tikimybių, priešingai nei kiti tyrimo metodai, todėl skaičiuojant metrikas šiais metodais kiekvieno defekto aptikimas buvo traktuojamas su tikimybe lygia 1. Atsižvelgus į PASCAL VOC 2012 mAP skaičiavimus (žr. 5 lentelę) pastebimas KNT metodų pranašumas prieš ne giliojo mokymosi metodus. Geriausių tyrimo modelių mAP reikšmė yra dešimtimis kartų didesnė už Hough apskritimų transformacijos algoritmą ar Suzuki algoritmą. Dėl tokio didelio skirtumo ne giliojo mokymosi metodai toliau tyrime nebebus nagrinėjami. Geriausius rezultatus parodė YOLOv4 modelis su 72,88% ir CenterNet HourGlass104 512x512 modelis su 68,13% mAP reikšmėmis. Prasčiausius rezultatus iš KNT davė SSD ResNet152 V1 FPN 640x640 (RetinaNet152) modelis su 11,33%.

5 lentelė. PASCAL VOC 2012 mAP metrika tyrime naudotų defektų atpažinimo metodų įvertinimui.

Defektų atpažinimo metodas	mAP
YOLOv4 modelis	72,88%
CenterNet HourGlass104 512x512 modelis	68,13%
Faster R-CNN Inception ResNet V2 640x640 modelis	31,12%
EfficientDet D0 512x512 modelis	30,5%
SSD ResNet152 V1 FPN 640x640 (RetinaNet152) modelis	11,33%
Suzuki algoritmas	3,97%
Hough apskritimų transformacija	2,78%

Nors YOLOv4 modelis pasiekė geriausią PASCAL VOC 2012 mAP rezultatą lyginant su kitais modeliais, bet MS COCO mAP metrikoje geriausią rezultatą parodė CenterNet HourGlass104 512x512 modelis su 27,20% pralenkęs YOLOv4 modelį su 24,75% (žr. 6 lentelę). Taip yra todėl nes su aukštesnėm IoU slenksčio reikšmėm CenterNet HourGlass104 512x512 modelis yra pranašesnis nei YOLOv4 modelis. Matome, kad mAP@.50 metrikoje YOLOv4 duoda geresnius rezultatus nei CenterNet HourGlass104 512x512 (71,03% prieš 67,27%), o padidinus IoU slenksčio reikšmę iki 0,75 mAP@.75 metrikoje CenterNet HourGlass104 512x512 modelis yra pranašesnis už YOLOv4 (15,58% prieš 7,86%). YOLOv4 modelio vidutinė aptiktų ribojančių stačiakampių IoU reikšmė yra 59,54% (žr. 7 lentelę). CenterNet HourGlass104 512x512 MS COCO mAP yra didesnis nepriklausomai nuo aptinkamo defekto dydžio, bet didesnis pranašumas aptinkant vidutinio dydžio defektus nei mažus defektus. Didelio dydžio defektų duomenyse nėra, todėl šiose lentelės eilutėse tuščia. AR metrikose fiksuojant vieną arba dešimt defektų aptikimų pranašesnis YOLOv4 modelis, tačiau fiksuojant 100 aptikimų nepriklausomai nuo aptikto defekto dydžio geresnius rezultatus duoda CenterNet HourGlass104 512x512 modelis. EfficientDet D0 512x512, SSD ResNet152 V1 FPN 640x640 (RetinaNet152) ir Faster R-CNN Inception ResNet V2 640x640 modeliai visose metrikose nusileidžia YOLOv4 ir CenterNet HourGlass104 512x512 modeliams.

Sprendžiant defektų aptikimo problemą yra svarbiau automatizuotas defektų aptikimas ir defektų kiekio nustatymas, o ne aptikto defekto ribojančio stačiakampio IoU reikšmės didinimas, todėl PASCAL VOC 2012 mAP metrika tyrimo kontekste yra svarbesnė už MS COCO 2014 mAP. Dėl šios priežasties tyrime buvo atliktas YOLOv4 modelio pagerinimo eksperimentas pridedant papildomą skaitmeninio vaizdo apdorojimo metodą (žr. 3 skyrių). Prie jau atliktų išankstinio vaizdo apdorojimo metodų (žr. 6.4 skyrių) buvo atliktas vienas papildomas metodas ir apskaičiuojamos painiavos matricos reikšmės kartu su vidutine IoU reikšme. Papildomai atlikti metodai buvo Canny kraštų aptikimo metodas, Otsu metodas ir Sobel briaunų radimo operatoriaus metodas. Canny kraštų aptikimo metodas buvo atliktas su slenksčio reikšmėmis lygiomis 75 ir 125. Geriausius rezultatus tarp pridėtinių vaizdo apdorojimo metodų painiavos matricos metrikose, iš jų įvedamose metrikose ir vidutiniame IoU parodė Canny kraštų aptikimo metodas (žr. 7 lentelę). Tačiau, joks papildomas vaizdo apdorojimo metodas nepagerino metrikų palyginus su jau atliktais išankstinio vaizdo apdorojimo metodais, kurie buvo skirti išvengti modelio persimokymo.

6 lentelė. MS COCO 2014 metrikos tyrimo modelio įvertinimams. TensorFlow Object Detection API modelių pavadinimai sutrumpinti, kad tilptų į lentelę. Pilni pavadinimai: CenterNet HourGlass104 512x512, Faster R-CNN Inception ResNet V2 640x640, EfficientDet D0 512x512, SSD ResNet152 V1 FPN 640x640 (RetinaNet152).

Modelis	YOLOv4	CenterNet	Faster R-CNN	EfficientDet	SSD
mAP	0.24756	0.27197	0.07530	0.12408	0.05233
mAP@.50	0.71034	0.67272	0.30096	0.29460	0.11729
mAP@.75	0.07860	0.15579	0.00737	0.07334	0.03982
mAP (small)	0.24476	0.26628	0.07862	0.11591	0.04568
mAP (medium)	0.35768	0.43663	0.12695	0.42459	0.25379
mAP (large)	-	-	-	-	-
AR@1	0.00909	0.00803	0.00612	0.00775	0.00478
AR@10	0.07770	0.07655	0.04095	0.06478	0.03846
AR@100	0.33502	0.36095	0.10937	0.15454	0.05770
AR@100 (small)	0.33336	0.35659	0.10039	0.14557	0.05118
AR@100 (medium)	0.39310	0.51379	0.42413	0.46896	0.28620
AR@100 (large)	-	-	-	-	-

7 lentelė. Darknet karkaso YOLOv4 modelių rezultatų metrikos pridedant papildomą skaitmeninio vaizdo apdorojimo metodą.

Apdorojimo metodas	Jokio	Canny	Otsu	Sobel
Precision	0,83	0,82	0,75	0,80
Recall	0,80	0,74	0,70	0,62
F1-score	0,82	0,78	0,72	0,70
TP	840	773	732	648
FP	173	165	244	160
FN	205	272	313	397
Vidutinis IoU	59,54%	59,06%	53,51%	56,88%

Išvados ir rekomendacijos

1. Visos konvoliucinių neuroninių tinklų tipo metodikos yra reikšmingai pranašesnės už ne dirbtinio intelekto metodikas atsižvelgiant į PASCAL VOC 2012 mAP metriką (žr. 5 lentelę, 37 psl.).

2. Geriausius PASCAL VOC 2012 mAP metrikos rezultatus pasiekė YOLOv4 modelis su 72,88% ir CenterNet HourGlass104 512x512 modelis su 68,13% mAP reikšmėmis (žr. 5 lentelę, 37 psl.). Likusių metodikų metrikų rezultatai yra ženkliai mažesni.

3. MS COCO 2014 mAP metrikoje geriausią rezultatą parodė CenterNet HourGlass104 512x512 modelis su 27,20% pralenkęs YOLOv4 modelį su 24,75% (žr. 6 lentelę, 38 psl.). Likusių metodikų metrikų rezultatai yra ženkliai mažesni.

4. Sprendžiant defektų aptikimo problemą atominės jėgos nuotraukose rekomenduojamas YOLOv4 modelis, nes automatizuotas defektų aptikimas ir defektų kiekio nustatymas (PASCAL VOC 2012 mAP) yra svarbiau už aptikto defekto ribojančio stačiakampio IoU reikšmės didinimą (MS COCO 2014 mAP).

5. YOLOv4 konvoliucinio neuroninio tinklo gaunamos metrikos nepagerėjo pridedant papildomą skaitmeninio vaizdo apdorojimo metodą (nagrinėtais Canny kraštų aptikimo, Otsu ar Sobel briaunų radimo operatoriaus metodų atvejais) duomenų ruošimo etape (žr. 7 lentelę, 38 psl.).

Ateities tyrimų gairės

Atsižvelgiant į atliktus darbus ir gautus rezultatus galimos šios tolimesnių tyrimų kryptys:

- Papildyti tyrimą defektų klasifikavimu. Defektus galima išskirti į $1/4$, $1/2$, $3/4$ ar pilno dydžio tipus (žr. 2.2 skyrių.).
- Papildyti tyrimą defektų klasterių aptikimu.
- Papildyti tyrimą atskiro KNT analize, kuris būtų apmokytas aptikti defektus tik klasteriuose su trimis ar daugiau defektais.
- Papildyti tyrimą skirtingų AJM kanalų analize.
- Papildyti tyrimą KNT analize, kuriam atrenkami kandidatai į defektus.
- Papildyti tyrimą KNT metrikų skaičiavimu naudojantis kryžmine koreliacija (angl. Cross-correlation).

Literatūros šaltiniai

[1] Mohammad Mahmudul Alam. Machine learning approach of automatic identification and counting of blood cells. Healthcare Technology Letters, 2019.

[2] Huitian Bai. Nanowire detection in afm images using deep learning. Microscopy and Microanalysis, 2021, 27.1: 54-64, 2021.

[3] Joseph Beton. Topostats – a program for automated tracing of biomolecules from afm images. Methods. 2021 Sep;193:68-79, 2021.

[4] Matplotlib biblioteka. https://matplotlib.org. Žiūrėta 2022-12-19.

[5] NumPy biblioteka. https://numpy.org. Žiūrėta 2022-12-14.

[6] OpenCV biblioteka. https://opencv.org. Žiūrėta 2022-12-14.

[7] Pandas biblioteka. https://pandas.pydata.org. Žiūrėta 2022-12-20.

[8] Pillow biblioteka. https://pillow.readthedocs.io. Žiūrėta 2022-12-20.

[9] Roboflow biblioteka. https://www.roboflow.com. Žiūrėta 2022-12-20.

[10] TensorFlow biblioteka. https://www.tensorflow.org. Žiūrėta 2022-12-14.

[11] OpenCV bibliotekos spalvų konvertavimo dokumentacija. https://docs.opencv.org/3.4/de/d25/imgproc_color_conversions.html. Žiūrėta 2021-09-06.

[12] Gerd Binnig. Atomic force microscope. Physical review letters, 56(9), 930, 1986.

[13] Pier Carlo Braga. Atomic force microscopy: Biomedical methods and applications. Humana Press, 2003.

[14] Kim Brogden. Antimicrobial peptides: pore formers or metabolic inhibitors in bacteria? Nat Rev Microbiol 3, 238–250, 2005.

[15] John Canny. A computational approach to edge detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 8, No. 6, Nov, 1986.

[16] COCO Detection Challenge. https://competitions.codalab.org/competitions/5181. Žiūrėta 2022-12-27.

[17] PASCAL VOC Challenge. http://host.robots.ox.ac.uk/pascal/VOC. Žiūrėta 2022-12-27.

[18] Tautminas Cibulskis. Algoritmų triukšmui šalinti duomenyse analizė. Vilniaus universitetas. Matematikos ir informatikos institutas, 2022.

[19] Emlyn Davies. Computer and machine vision: Theory, algorithms and practicalities. Academic Press, 1990.

[20] Gintaras Dreižas. Defektų įtakos lipidinių membranų laidumui modeliavimas baigtinių elementų metodu. Vilniaus universitetas. Matematikos ir informatikos institutas, 2017.

[21] Borko Furht. Digital image processing: Practical approach. SpringerBriefs in Computer Science, 2018.

[22] Rafael Gonzales. Digital image processing. Prentice Hall PTR, 2008.

[23] Ian Goodfellow. Deep learning. The MIT Press, 2015.

[24] Prathima Guruprasad. Overview of different thresholding methods in image processing. TEQIP Sponsored 3rd National Conference on ETACC, 2020.

[25] imgaug biblioteka. https://pillow.readthedocs.io. Žiūrėta 2022-12-20.

[26] Rahul Kumar. Machine learning quick reference: Quick and essential machine learning hacks for training smart data models. Packt Publiching, 2019.

[27] Brendan Marsh. The hessian blob algorithm: Precise particle detection in atomic force microscopy imagery. Scientific Reports, 2018.

[28] Vytenis Navalinskas. Defektų atpažinimas biologinėse membranose. Vilniaus universitetas. Matematikos ir informatikos institutas, 2020.

[29] Nobuyuki Otsu. A threshold selection method from gray-level histograms. IEEE Transactions on Systems, Man, and Cybernetics, 1979.

[30] Simon Just Kjeldgaard Pedersen. Circular hough transform. Aalborg University. Vision, Graphics and Interactive Systems, 2007.

[31] Tomas Raila. Ai-based atomic force microscopy image analysis allows to predict electrochemical impedance spectra of defects in tethered bilayer membranes. Scientific reports 12.1 (2022): 1-11, 2022.

[32] Darknet repozitorija. https://github.com/AlexeyAB/darknet. Žiūrėta 2022-06-16.

[33] LabelImg repozitorija. https://github.com/tzutalin/labelImg. Žiūrėta 2022-06-16.

[34] Object Detection Metrics GitHub repozitorija. https://github.com/rafaelpadilla/review_object_detection_metrics. Žiūrėta 2022-12-20.

[35] GitHub repozitorijoje esantis objektų aptikimo kodas. https://github.com/abdelrahman-gaber/tf2-object-detection-api-tutorial/blob/master/detect_objects.py. Žiūrėta 2022-12-20.

[36] GitHub repozitorijoje esantis Tensorflow Object Detection CSV formatą generuojantis kodas. https://raw.githubusercontent.com/datitran/raccoon_dataset/master/xml_to_csv.py. Žiūrėta 2022-12-20.

[37] GitHub repozitorijoje esantis Tensorflow TFRecord formatą generuojantis kodas. https://github.com/abdelrahman-gaber/tf2-object-detection-api-tutorial/blob/master/data_gen/generate_tfrecord.py. Žiūrėta 2022-12-20.

[38] Linda Shapiro. Computer vision. Prentice Hall PTR, 2001.

[39] Javier Sotres. Enabling autonomous scanning probe microscopy imaging of single molecules with deep learning. Nanoscale, 2021, 13.20: 9193-9203., 2021.

[40] William Stillwell. An introduction to biological membranes. Elsevier Academic Press, 2013.

[41] Satoshi Suzuki. Topological structural analysis of digitized binary images by border following. Computer Vision, Graphics and Image Processing, 32-46, 1985.

[42] Vipin Tyagi. Understanding digital image processing. CRC Press, 2018.

[43] Gintaras Valincius. Electrochemical impedance spectroscopy of tethered bilayer membranes. Langmuir. 2012 Jan 10;28(1):977-90, 2012.

[44] David Vernon. Machine vision. Prentice-Hall, 1991.

[45] Vikipedija. https://en.wikipedia.org/wiki/Atomic_force_microscopy. Žiūrėta 2022-12-14.

[46] Vikipedija. https://lt.wikipedia.org/wiki/Plazminė_membrana. Žiūrėta 2022-12-12.

[47] Igor Vilghelm. Automatizuoti biologinių membranų defektų atpažinimo algoritmai atominės jėgos mikroskopo nuotraukose. Vilniaus universitetas. Matematikos ir informatikos institutas, 2021.

[48] Helen Watson. Biological membranes. Essays in biochemistry, 59, 43–69, 2015.

[49] Virendra Kumar Yadav. Approach to accurate circle detection: Circular hough transform and local maxima concept. International Conference on Electronics and Communication Systems (lCECS -2014), Feb.13 -14, 2014.

[50] Chao-Chao Zhang. Edge detection based on improved sobel operator. 2016 International Conference on Computer Engineering and Information Systems (CEIS-16), 2016.

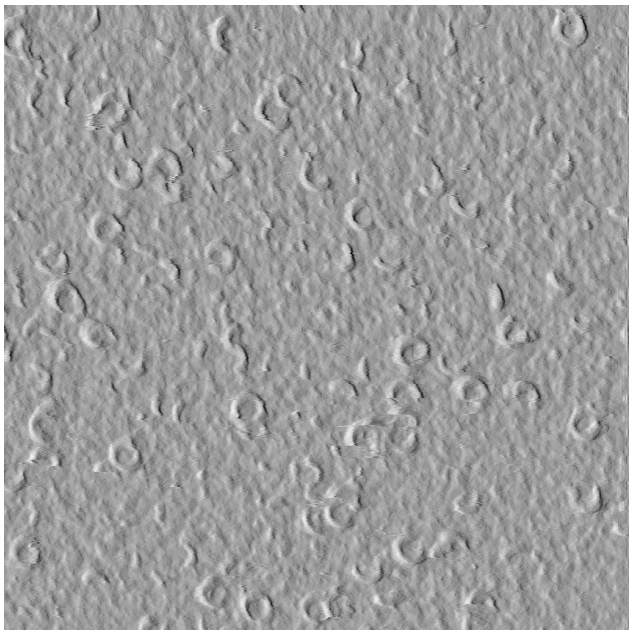
[51] Junkang Zhang. Cancer cells detection in phase-contrast microscopy images based on faster r-cnn. 9th International Symposium on Computational Intelligence and Design, 2016.

Priedai

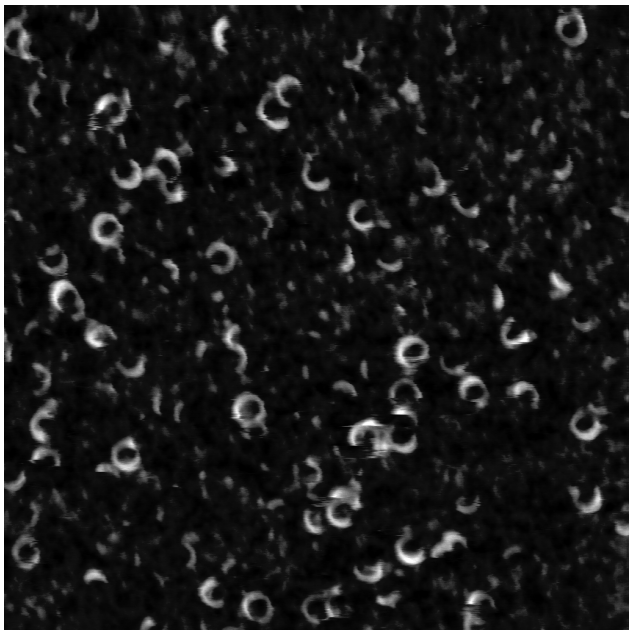
Magistro baigiamajame darbe yra 3 priedai:

- A priedas. Pateikiami atominės jėgos mikroskopijos kanalai.
- B priedas. Pateikiami išankstinio vaizdų apdorojimo metodai.
- C priedas. Pateikiami naudotų objektų sužymėjimo formatų pavyzdžiai.
- D priedas. Pateikiama naudoto YOLOv4 konvoliucinio neuroninio tinklo architektūra.
- E priedas. Pateikiami tyrimo defektų aptikimo metodų detekcijų pavyzdžiai.

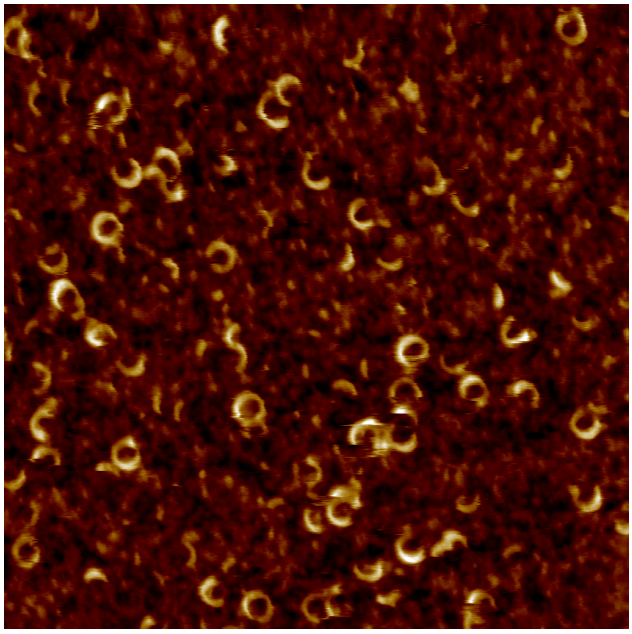
A. Atominės jėgos mikroskopijos kanalai



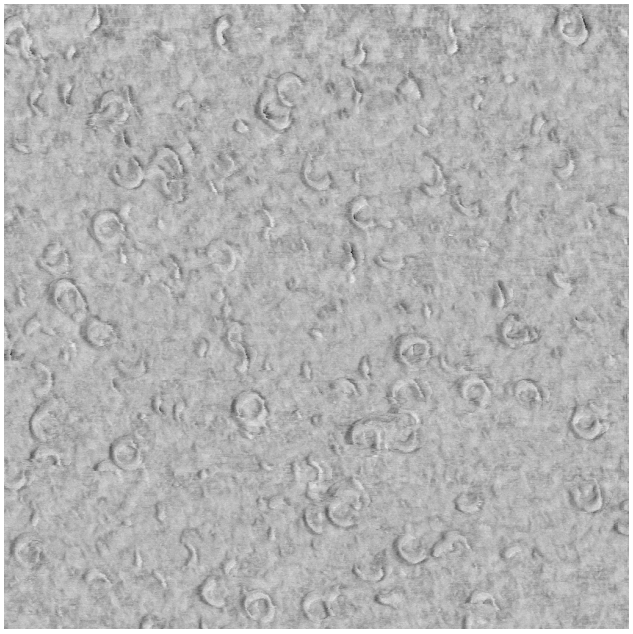
(a) Amplitudės kanalo vaizdas.



(b) Aukščio kanalo vaizdas.



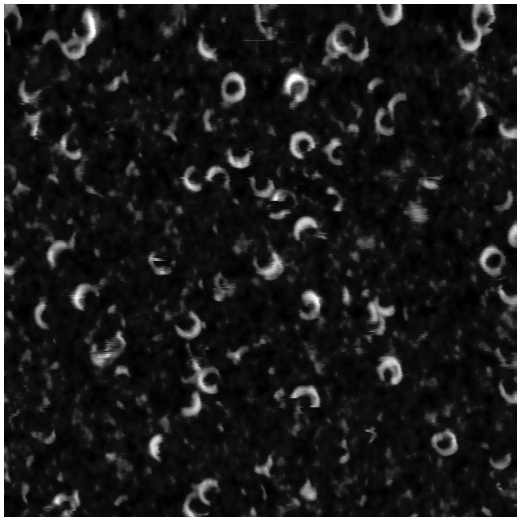
(c) Aukščio kanalo vaizdas konvertuotas į spalvotą.



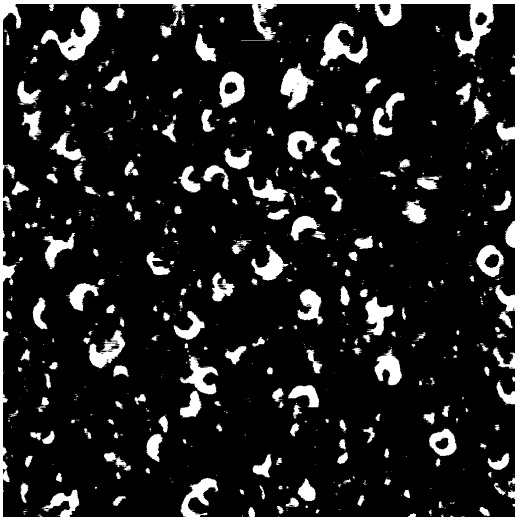
(d) Fazės kanalo vaizdas.

5 pav. Skirtingų atominės jėgos mikroskopijos kanalų vaizdai.

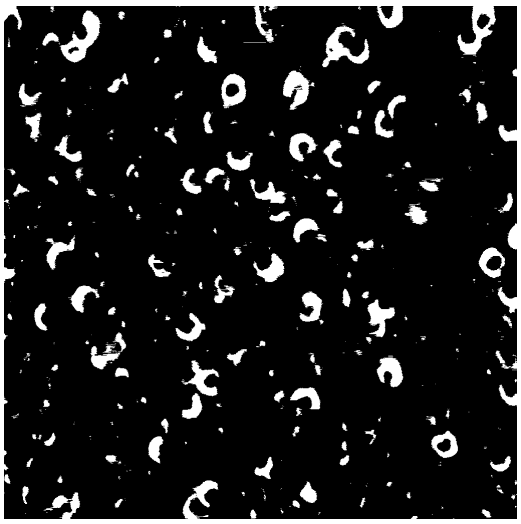
B. Išankstinio vaizdų apdorojimo metodai



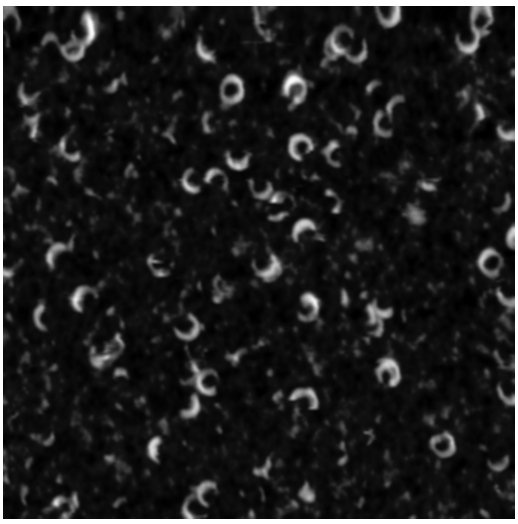
(a) Aukščio kanalo vaizdas.



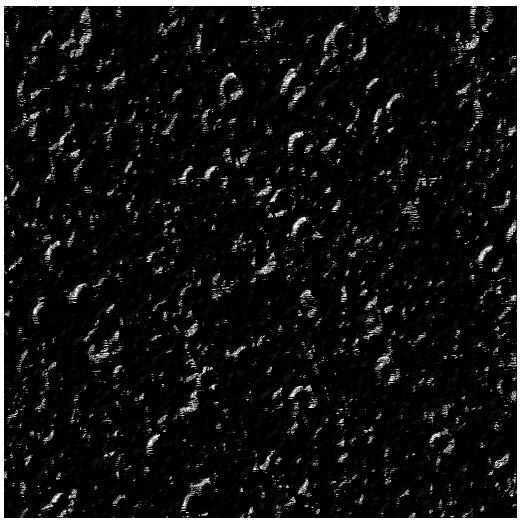
(b) Slenkstinis triukšmo filtras (T = 50).



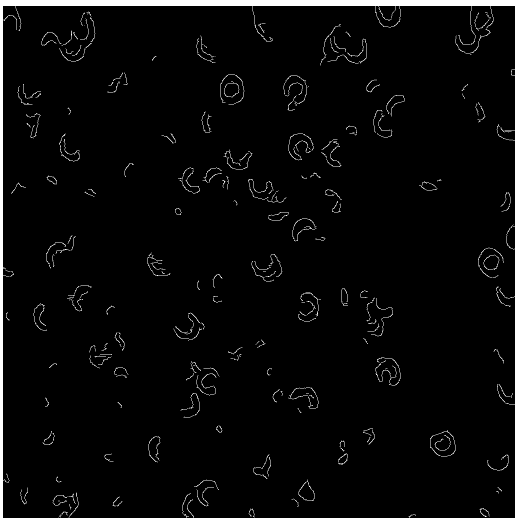
(c) Otsu segmentavimo metodas.



(d) Gauso glodinimo filtras.



(e) Sobel briaunų radimo operatorius.



(f) Canny kraštų aptikimo metodas.

6 pav. Išankstinio vaizdų apdorojimo metodų pavyzdžiai.

C. Naudotų objektų sužymėjimo formatų pavyzdžiai

```
<annotation>
        <folder>heights</folder>
        <filename>p1.png</filename>
        <path>/example/p1.png</path>
        <source>
                <database>Unknown</database>
        </source>
        <size>
                <width>1024</width>
                <height>1024</height>
                <depth>3</depth>
        </size>
        <segmented>0</segmented>
        <object>
                <name>defect</name>
                <pose>Unspecified</pose>
                <truncated>0</truncated>
                <dificult>0</dificult>
                <occluded>0</occluded>
                <bndbox>
                        <xmin>-12</xmin>
                        <xmax>31</xmax>
                        <ymin>38</ymin>
                        <ymax>81</ymax>
                </bndbox>
        </object>
</annotation>
```

7 pav. Pascal VOC XML failo pavyzdys.

```
0 0.652344 0.554688 0.058594 0.062500
0 0.908691 0.035645 0.057617 0.063477
0 0.349609 0.403320 0.052734 0.054688
```

8 pav. YOLO Darknet TXT failo pavyzdys. Labels faile atskirose eilutėse surašomi objektai.

```
filename,width,height,class,xmin,ymin,xmax,ymax
01.png,500,375,defect,111,144,134,174
01.png,500,375,defect,178,84,230,143
07.png,500,466,defect,115,139,180,230
07.png,500,466,defect,174,156,201,219
07.png,500,466,defect,197,177,231,227
07.png,500,466,defect,247,124,294,203
07.png,500,466,defect,280,127,337,208
07.png,500,466,defect,336,148,387,223
07.png,500,466,defect,375,152,410,219
```

9 pav. Tensorflow Object Detection CSV failo pavyzdys.

D. Tyrime naudoto YOLOv4 konvoliucinio neuroninio tinklo architektūra

8 lentelė. Tyrime naudoto YOLOv4 modelio architektūros sluoksniai. Arctitektūrą galima rasti tyrimo praktinės dalies Darknet direktorijoje esančiame modifikuotame konfigūracijos faile arba autoriaus sukurto def_custom įrankio išvestyje.

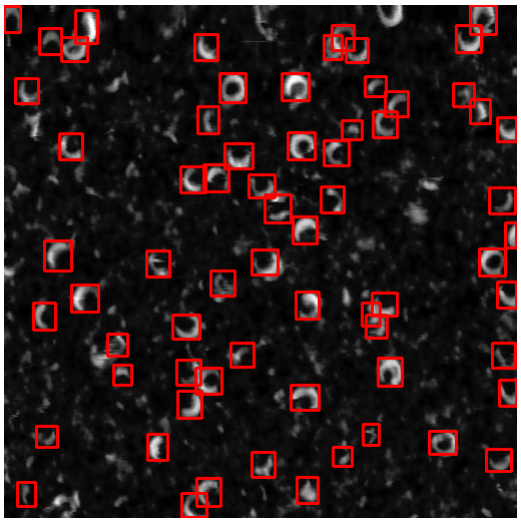
Nr	Sluoksnis	Filtrai	Dydis	Įvesties dydis	Išvesties dydis
0	Konvoliucinis	32	3 x 3	608 x 608 x 3	608 x 608 x 32
1	Konvoliucinis	64	3 x 3	608 x 608 x 32	304 x 304 x 64
2	Konvoliucinis	64	1 x 1	304 x 304 x 64	304 x 304 x 64
3	Maršrutinis	Paimama 1 sluoksnio išvestis			304 x 304 x 64
4	Konvoliucinis	64	1 x 1	304 x 304 x 64	304 x 304 x 64
5	Konvoliucinis	32	1 x 1	304 x 304 x 64	304 x 304 x 32
6	Konvoliucinis	64	3 x 3	304 x 304 x 32	304 x 304 x 64
7	Sutrumpinimo: 4				304 x 304 x 64
8	Konvoliucinis	64	1 x 1	304 x 304 x 64	304 x 304 x 64
9	Maršrutinis	Paimamos 8 ir 2 sluoksnio išvestys			304 x 304 x 128
10	Konvoliucinis	64	1 x 1	304 x 304 x 128	304 x 304 x 64
11	Konvoliucinis	128	3 x 3	304 x 304 x 64	152 x 152 x 128
12	Konvoliucinis	64	1 x 1	152 x 152 x 128	152 x 152 x 64
13	Maršrutinis	Paimama 11 sluoksnio išvestis			152 x 152 x 128
14	Konvoliucinis	64	1 x 1	152 x 152 x 128	152 x 152 x 64
15	Konvoliucinis	64	1 x 1	152 x 152 x 64	152 x 152 x 64
16	Konvoliucinis	64	3 x 3	152 x 152 x 64	152 x 152 x 64
17	Sutrumpinimo: 14				152 x 152 x 64
18	Konvoliucinis	64	1 x 1	152 x 152 x 64	152 x 152 x 64
19	Konvoliucinis	64	3 x 3	152 x 152 x 64	152 x 152 x 64
20	Sutrumpinimo: 17				152 x 152 x 64
21	Konvoliucinis	64	1 x 1	152 x 152 x 64	152 x 152 x 64
22	Maršrutinis	Paimamos 21 ir 12 sluoksnio išvestys			152 x 152 x 128
23	Konvoliucinis	128	1 x 1	152 x 152 x 128	152 x 152 x 128
24	Konvoliucinis	256	3 x 3	152 x 152 x 128	76 x 76 x 256
25	Konvoliucinis	128	1 x 1	76 x 76 x 256	76 x 76 x 128
26	Maršrutinis	Paimama 24 sluoksnio išvestis			76 x 76 x 256
27	Konvoliucinis	128	1 x 1	76 x 76 x 256	76 x 76 x 128
28	Konvoliucinis	128	1 x 1	76 x 76 x 128	76 x 76 x 128
29	Konvoliucinis	128	3 x 3	76 x 76 x 128	76 x 76 x 128
30	Sutrumpinimo: 27				76 x 76 x 128
31	Konvoliucinis	128	1 x 1	76 x 76 x 128	76 x 76 x 128
32	Konvoliucinis	128	3 x 3	76 x 76 x 128	76 x 76 x 128
33	Sutrumpinimo: 30				76 x 76 x 128
34	Konvoliucinis	128	1 x 1	76 x 76 x 128	76 x 76 x 128
35	Konvoliucinis	128	3 x 3	76 x 76 x 128	76 x 76 x 128

36	Sutrumpinimo: 33				76 x 76 x 128
37	Konvoliucinis	128	1 x 1	76 x 76 x 128	76 x 76 x 128
38	Konvoliucinis	128	3 x 3	76 x 76 x 128	76 x 76 x 128
39	Sutrumpinimo: 36				76 x 76 x 128
40	Konvoliucinis	128	1 x 1	76 x 76 x 128	76 x 76 x 128
41	Konvoliucinis	128	3 x 3	76 x 76 x 128	76 x 76 x 128
42	Sutrumpinimo: 39				76 x 76 x 128
43	Konvoliucinis	128	1 x 1	76 x 76 x 128	76 x 76 x 128
44	Konvoliucinis	128	3 x 3	76 x 76 x 128	76 x 76 x 128
45	Sutrumpinimo: 42				76 x 76 x 128
46	Konvoliucinis	128	1 x 1	76 x 76 x 128	76 x 76 x 128
47	Konvoliucinis	128	3 x 3	76 x 76 x 128	76 x 76 x 128
48	Sutrumpinimo: 45				76 x 76 x 128
49	Konvoliucinis	128	1 x 1	76 x 76 x 128	76 x 76 x 128
50	Konvoliucinis	128	3 x 3	76 x 76 x 128	76 x 76 x 128
51	Sutrumpinimo: 48				76 x 76 x 128
52	Konvoliucinis	128	1 x 1	76 x 76 x 128	76 x 76 x 128
53	Maršrutinis	Paimamos 52 ir 25 sluoksnio išvestys			76 x 76 x 256
54	Konvoliucinis	256	1 x 1	76 x 76 x 256	76 x 76 x 256
55	Konvoliucinis	512	3 x 3	76 x 76 x 256	38 x 38 x 512
56	Konvoliucinis	256	1 x 1	38 x 38 x 512	38 x 38 x 256
57	Maršrutinis	Paimama 55 sluoksnio išvestis			38 x 38 x 512
58	Konvoliucinis	256	1 x 1	38 x 38 x 512	38 x 38 x 256
59	Konvoliucinis	256	1 x 1	38 x 38 x 256	38 x 38 x 256
60	Konvoliucinis	256	3 x 3	38 x 38 x 256	38 x 38 x 256
61	Sutrumpinimo: 58				38 x 38 x 256
62	Konvoliucinis	256	1 x 1	38 x 38 x 256	38 x 38 x 256
63	Konvoliucinis	256	3 x 3	38 x 38 x 256	38 x 38 x 256
64	Sutrumpinimo: 61				38 x 38 x 256
65	Konvoliucinis	256	1 x 1	38 x 38 x 256	38 x 38 x 256
66	Konvoliucinis	256	3 x 3	38 x 38 x 256	38 x 38 x 256
67	Sutrumpinimo: 64				38 x 38 x 256
68	Konvoliucinis	256	1 x 1	38 x 38 x 256	38 x 38 x 256
69	Konvoliucinis	256	3 x 3	38 x 38 x 256	38 x 38 x 256
70	Sutrumpinimo: 67				38 x 38 x 256
71	Konvoliucinis	256	1 x 1	38 x 38 x 256	38 x 38 x 256
72	Konvoliucinis	256	3 x 3	38 x 38 x 256	38 x 38 x 256
73	Sutrumpinimo: 70				38 x 38 x 256
74	Konvoliucinis	256	1 x 1	38 x 38 x 256	38 x 38 x 256
75	Konvoliucinis	256	3 x 3	38 x 38 x 256	38 x 38 x 256
76	Sutrumpinimo: 73				38 x 38 x 256
77	Konvoliucinis	256	1 x 1	38 x 38 x 256	38 x 38 x 256
78	Konvoliucinis	256	3 x 3	38 x 38 x 256	38 x 38 x 256
79	Sutrumpinimo: 76				38 x 38 x 256

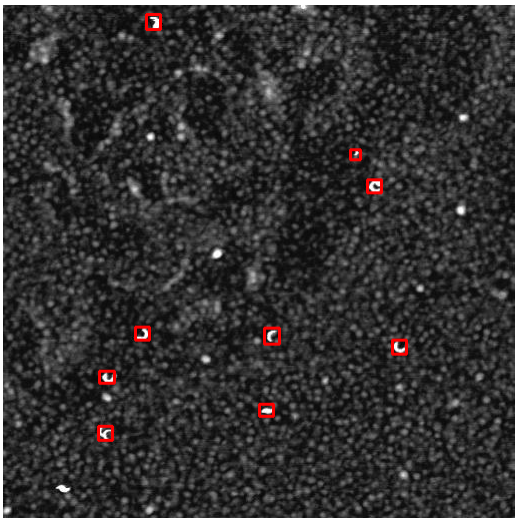
80	Konvoliucinis	256	1 x 1	38 x 38 x 256	38 x 38 x 256
81	Konvoliucinis	256	3 x 3	38 x 38 x 256	38 x 38 x 256
82	Sutrumpinimo: 79				38 x 38 x 256
83	Konvoliucinis	256	1 x 1	38 x 38 x 256	38 x 38 x 256
84	Maršrutinis	Paimamos 83 ir 56 sluoksnio išvestys			38 x 38 x 512
85	Konvoliucinis	512	1 x 1	38 x 38 x 512	38 x 38 x 512
86	Konvoliucinis	1024	3 x 3	38 x 38 x 512	19 x 19 x1024
87	Konvoliucinis	512	1 x 1	19 x 19 x1024	19 x 19 x 512
88	Maršrutinis	Paimama 86 sluoksnio išvestis			19 x 19 x 1024
89	Konvoliucinis	512	1 x 1	19 x 19 x 1024	19 x 19 x 512
90	Konvoliucinis	512	1 x 1	19 x 19 x 512	19 x 19 x 512
91	Konvoliucinis	512	3 x 3	19 x 19 x 512	19 x 19 x 512
92	Sutrumpinimo: 89				19 x 19 x 512
93	Konvoliucinis	512	1 x 1	19 x 19 x 512	19 x 19 x 512
94	Konvoliucinis	512	3 x 3	19 x 19 x 512	19 x 19 x 512
95	Sutrumpinimo: 92				19 x 19 x 512
96	Konvoliucinis	512	1 x 1	19 x 19 x 512	19 x 19 x 512
97	Konvoliucinis	512	3 x 3	19 x 19 x 512	19 x 19 x 512
98	Sutrumpinimo: 95				19 x 19 x 512
99	Konvoliucinis	512	1 x 1	19 x 19 x 512	19 x 19 x 512
100	Konvoliucinis	512	3 x 3	19 x 19 x 512	19 x 19 x 512
101	Sutrumpinimo: 98				19 x 19 x 512
102	Konvoliucinis	512	1 x 1	19 x 19 x 512	19 x 19 x 512
103	Maršrutinis	Paimamos 102 ir 87 sluoksnio išvestys			19 x 19 x 1024
104	Konvoliucinis	1024	1 x 1	19 x 19 x1024	19 x 19 x1024
105	Konvoliucinis	512	1 x 1	19 x 19 x1024	19 x 19 x 512
106	Konvoliucinis	1024	3 x 3	19 x 19 x 512	19 x 19 x1024
107	Konvoliucinis	512	1 x 1	19 x 19 x1024	19 x 19 x 512
108	Sutelkimo		5x 5	19 x 19 x 512	19 x 19 x 512
109	Maršrutinis	Paimama 107 sluoksnio išvestis			19 x 19 x 512
110	Sutelkimo		9x 9	19 x 19 x 512	19 x 19 x 512
111	Maršrutinis	Paimama 107 sluoksnio išvestis			19 x 19 x 512
112	Sutelkimo		13x13	19 x 19 x 512	19 x 19 x 512
113	Maršrutinis	Paimamos 112, 110, 108, 107 sluoksnių išvestys			19 x 19 x2048
114	Konvoliucinis	512	1 x 1	19 x 19 x2048	19 x 19 x 512
115	Konvoliucinis	1024	3 x 3	19 x 19 x 512	19 x 19 x1024
116	Konvoliucinis	512	1 x 1	19 x 19 x1024	19 x 19 x 512
117	Konvoliucinis	256	1 x 1	19 x 19 x 512	19 x 19 x 256
118	Praplėtimo		2	19 x 19 x 256	38 x 38 x 256
119	Maršrutinis	Paimama 85 sluoksnio išvestis			38 x 38 x 512
120	Konvoliucinis	256	1 x 1	38 x 38 x 512	38 x 38 x 256
121	Maršrutinis	Paimamos 120 ir 118 sluoksnių išvestys			38 x 38 x 512
122	Konvoliucinis	256	1 x 1	38 x 38 x 512	38 x 38 x 256
123	Konvoliucinis	512	3 x 3	38 x 38 x 256	38 x 38 x 512

124	Konvoliucinis	256	1 x 1	38 x 38 x 512	38 x 38 x 256
125	Konvoliucinis	512	3 x 3	38 x 38 x 256	38 x 38 x 512
126	Konvoliucinis	256	1 x 1	38 x 38 x 512	38 x 38 x 256
127	Konvoliucinis	128	1 x 1	38 x 38 x 256	38 x 38 x 128
128	Praplėtimo		2x	38 x 38 x 128	76 x 76 x 128
129	Maršrutinis	Paimama 54 sluoksnio išvestis			76 x 76 x 256
130	Konvoliucinis	128	1 x 1	76 x 76 x 256	76 x 76 x 128
131	Maršrutinis	Paimamos 130 ir 128 sluoksnio išvestys			76 x 76 x 256
132	Konvoliucinis	128	1 x 1	76 x 76 x 256	76 x 76 x 128
133	Konvoliucinis	256	3 x 3	76 x 76 x 128	76 x 76 x 256
134	Konvoliucinis	128	1 x 1	76 x 76 x 256	76 x 76 x 128
135	Konvoliucinis	256	3 x 3	76 x 76 x 128	76 x 76 x 256
136	Konvoliucinis	128	1 x 1	76 x 76 x 256	76 x 76 x 128
137	Konvoliucinis	256	3 x 3	76 x 76 x 128	76 x 76 x 256
138	Konvoliucinis	255	1 x 1	76 x 76 x 256	76 x 76 x 255
139	YOLO				
140	Maršrutinis	Paimama 136 sluoksnio išvestis			76 x 76 x 128
141	Konvoliucinis	256	3 x 3	76 x 76 x 128	38 x 38 x 256
142	Maršrutinis	Paimamos 141 ir 126 sluoksnių išvestys			38 x 38 x 512
143	Konvoliucinis	256	1 x 1	38 x 38 x 512	38 x 38 x 256
144	Konvoliucinis	512	3 x 3	38 x 38 x 256	38 x 38 x 512
145	Konvoliucinis	256	1 x 1	38 x 38 x 512	38 x 38 x 256
146	Konvoliucinis	512	3 x 3	38 x 38 x 256	38 x 38 x 512
147	Konvoliucinis	256	1 x 1	38 x 38 x 512	38 x 38 x 256
148	Konvoliucinis	512	3 x 3	38 x 38 x 256	38 x 38 x 512
149	Konvoliucinis	255	1 x 1	38 x 38 x 512	38 x 38 x 255
150	YOLO				
151	Maršrutinis	Paimama 147 sluoksnio išvestis			38 x 38 x 256
152	Konvoliucinis	512	3 x 3	38 x 38 x 256	19 x 19 x 512
153	Maršrutinis	Paimamos 152 ir 116 sluoksnių išvestys			19 x 19 x 1024
154	Konvoliucinis	512	1 x 1	19 x 19 x1024	19 x 19 x 512
155	Konvoliucinis	1024	3 x 3	19 x 19 x 512	19 x 19 x 1024
156	Konvoliucinis	512	1 x 1	19 x 19 x1024	19 x 19 x 512
157	Konvoliucinis	1024	3 x 3	19 x 19 x 512	19 x 19 x 1024
158	Konvoliucinis	512	1 x 1	19 x 19 x1024	19 x 19 x 512
159	Konvoliucinis	1024	3 x 3	19 x 19 x 512	19 x 19 x 1024
160	Konvoliucinis	255	1 x 1	19 x 19 x1024	19 x 19 x 255
161	YOLO				

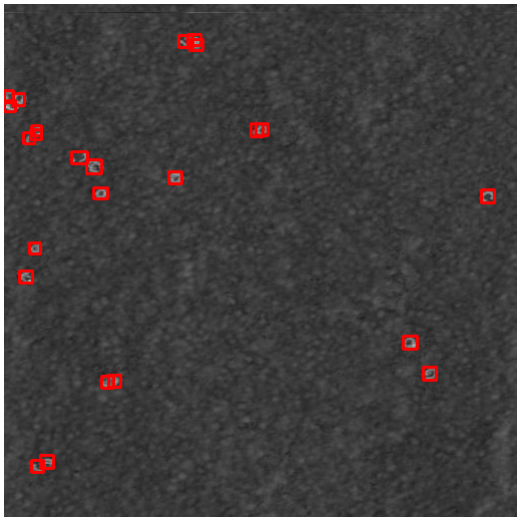
E. Tyrimo defektų aptikimo metodų detekcijų pavyzdžiai



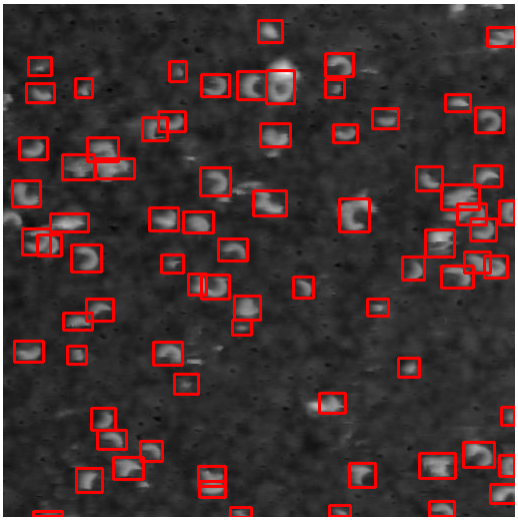
(a) 1 membranos vaizdas.



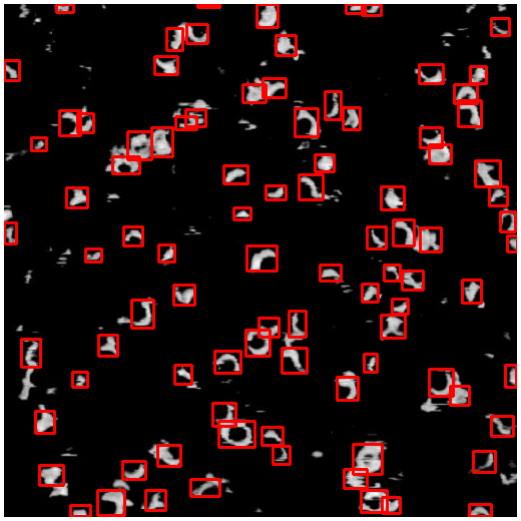
(b) 2 membranos vaizdas.



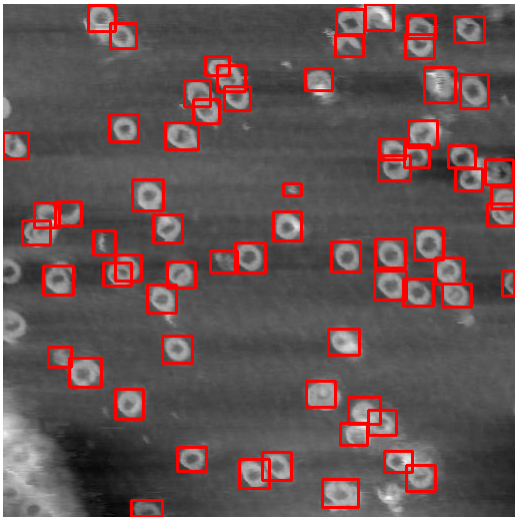
(c) 3 membranos vaizdas.



(d) 4 membranos vaizdas.

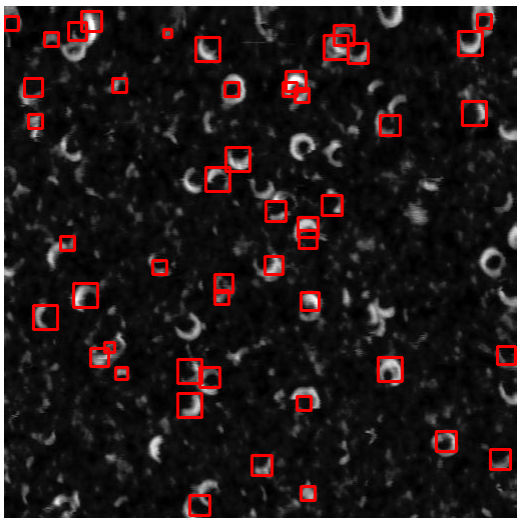


(e) 5 membranos vaizdas.

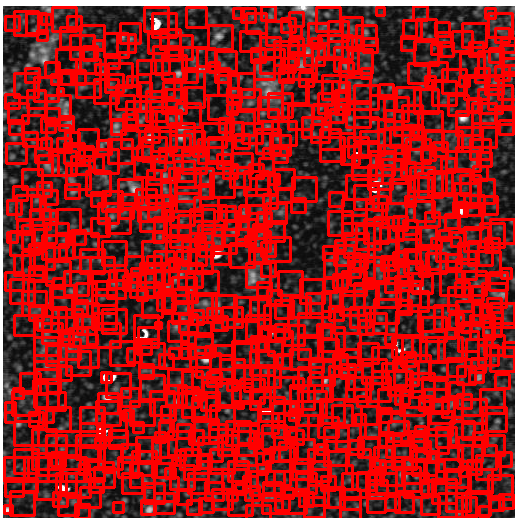


(f) 6 membranos vaizdas.

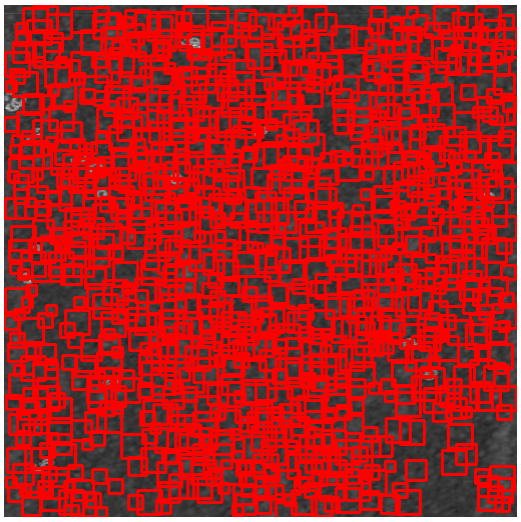
10 pav. YOLOv4 modelio defektų aptikimas testavimo aibės vaizduose.



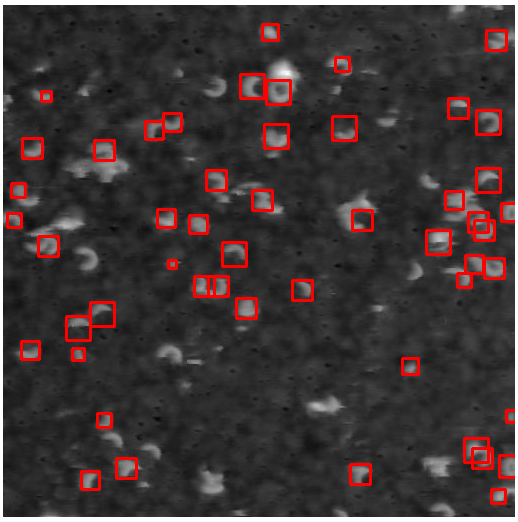
(a) 1 membranos vaizdas.



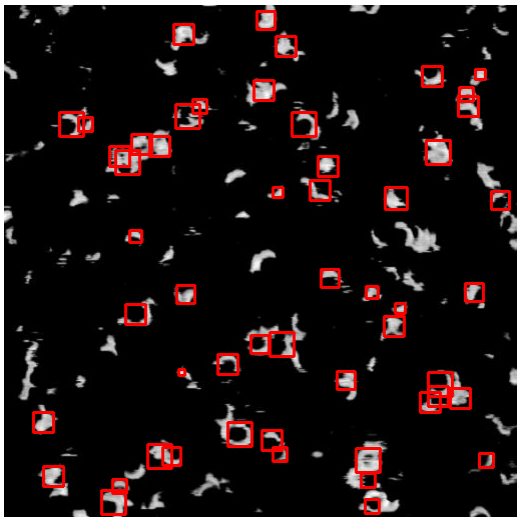
(b) 2 membranos vaizdas.



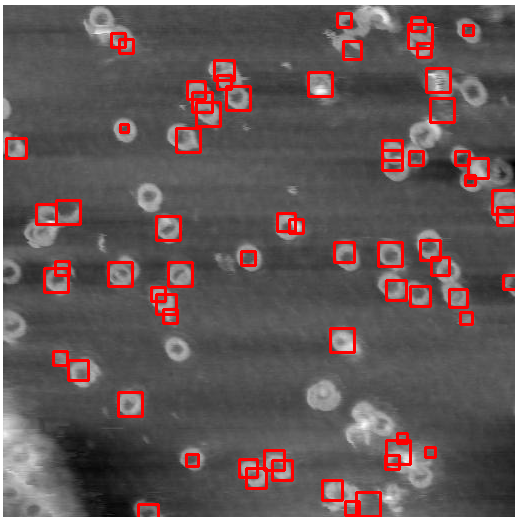
(c) 3 membranos vaizdas.



(d) 4 membranos vaizdas.

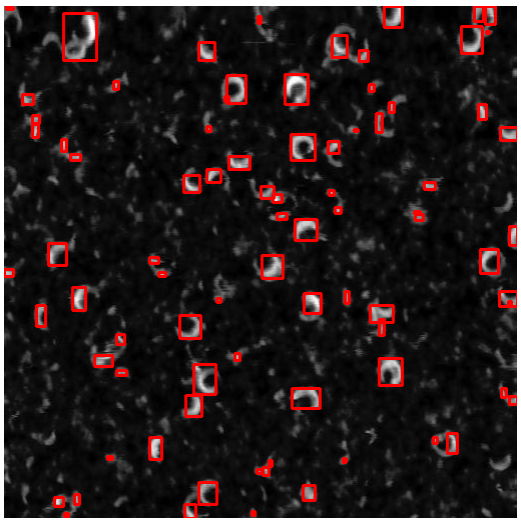


(e) 5 membranos vaizdas.

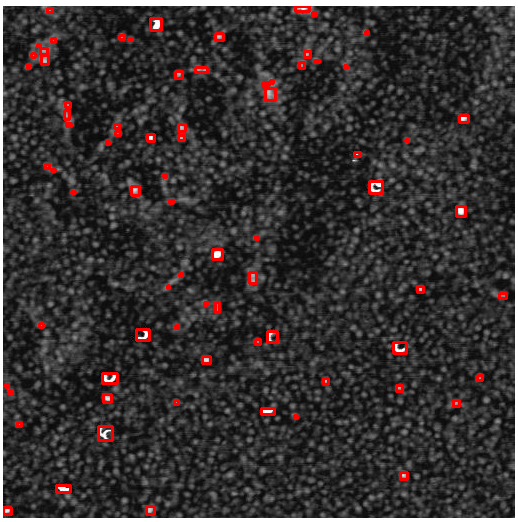


(f) 6 membranos vaizdas.

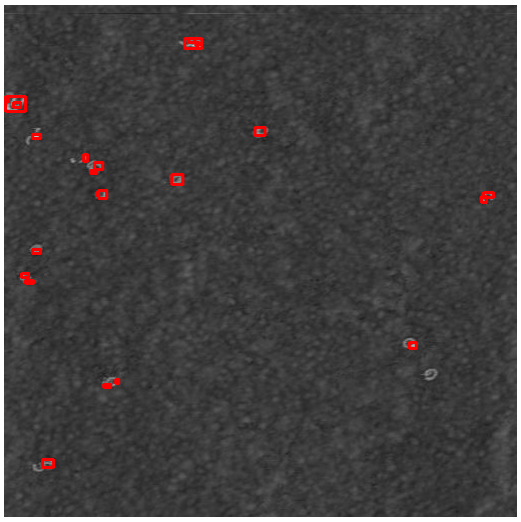
11 pav. Hough apskritimų transformacijos algoritmo defektų aptikimas testavimo aibės vaizduose.



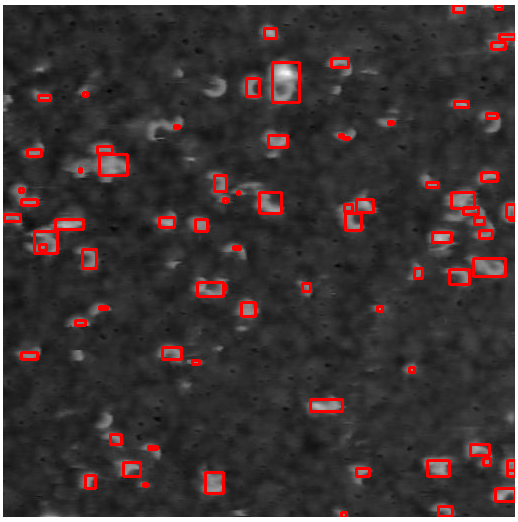
(a) 1 membranos vaizdas.



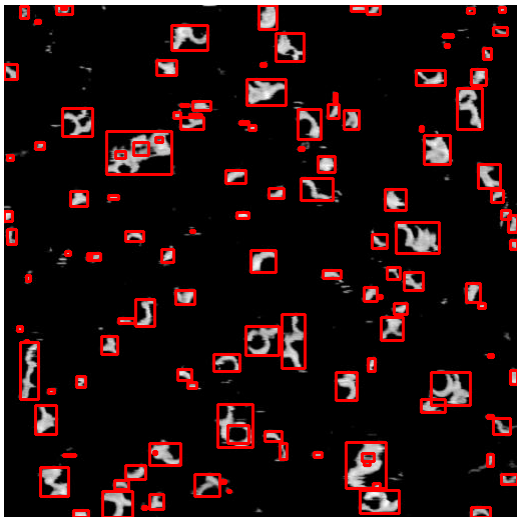
(b) 2 membranos vaizdas.



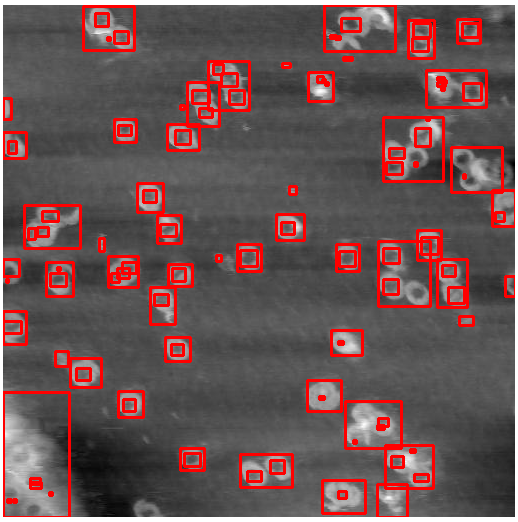
(c) 3 membranos vaizdas.



(d) 4 membranos vaizdas.

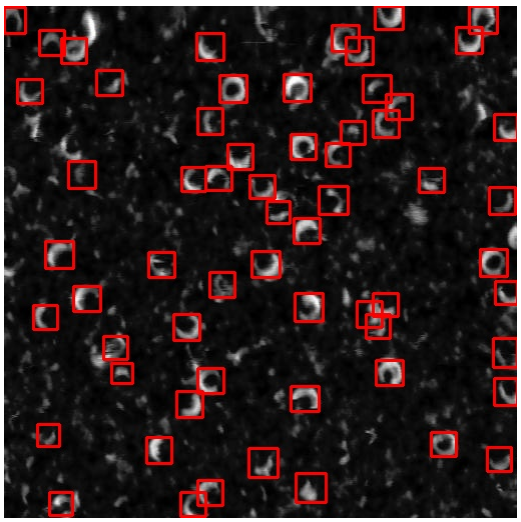


(e) 5 membranos vaizdas.

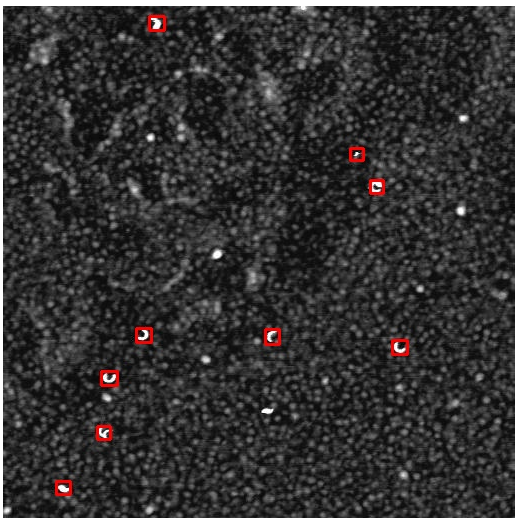


(f) 6 membranos vaizdas.

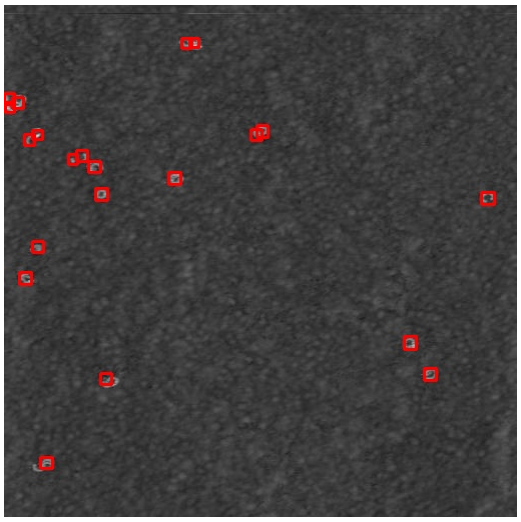
12 pav. Suzuki algoritmo defektų aptikimas testavimo aibės vaizduose.



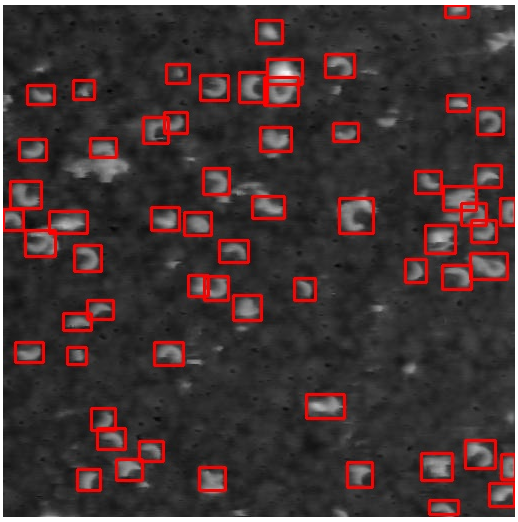
(a) 1 membranos vaizdas.



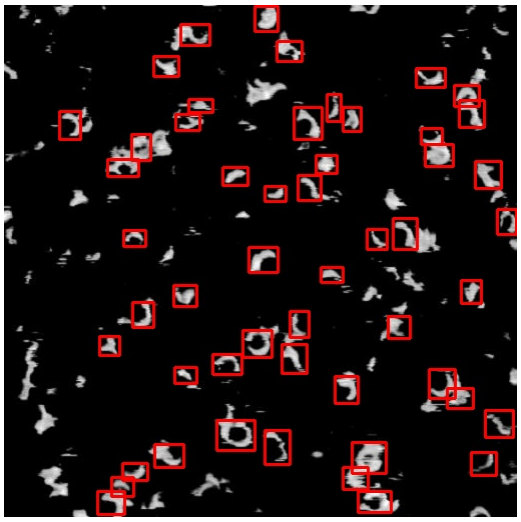
(b) 2 membranos vaizdas.



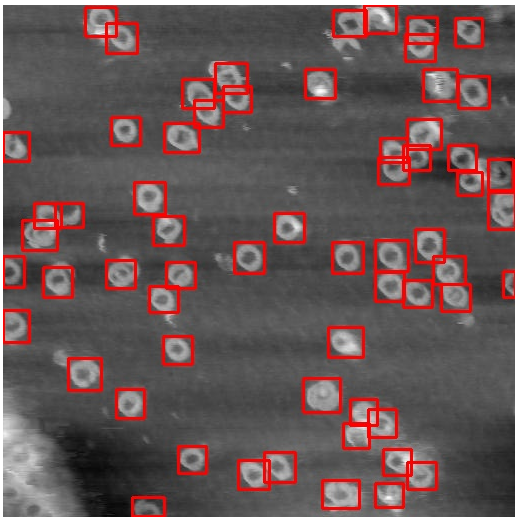
(c) 3 membranos vaizdas.



(d) 4 membranos vaizdas.

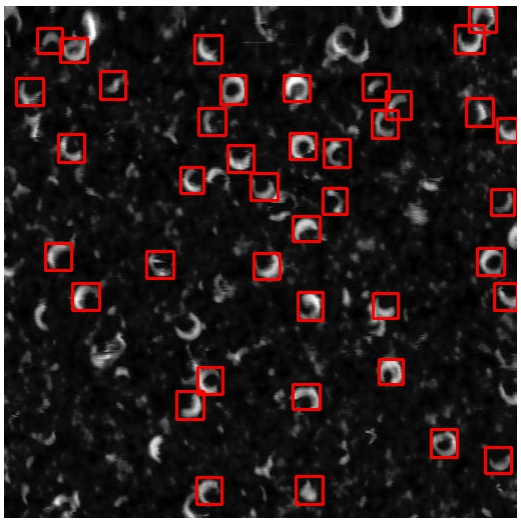


(e) 5 membranos vaizdas.

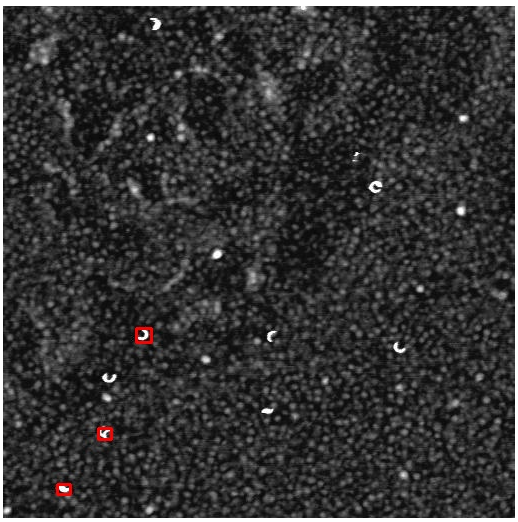


(f) 6 membranos vaizdas.

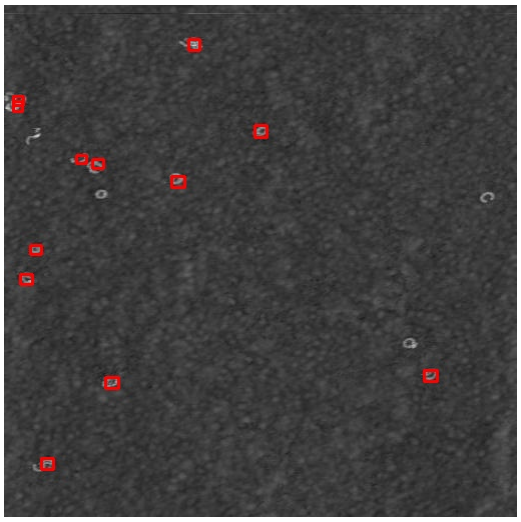
13 pav. CenterNet HourGlass104 512x512 modelio defektų aptikimas testavimo aibės vaizduose.



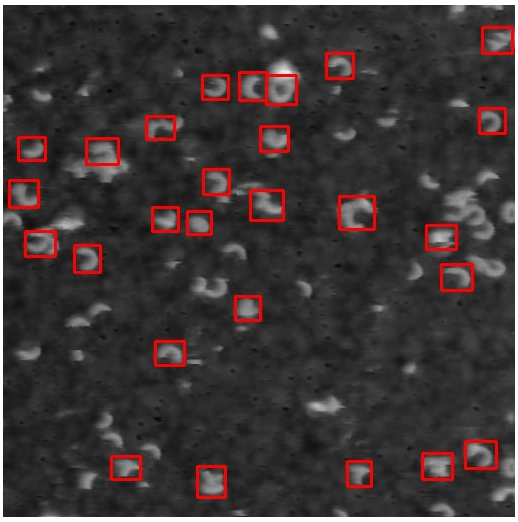
(a) 1 membranos vaizdas.



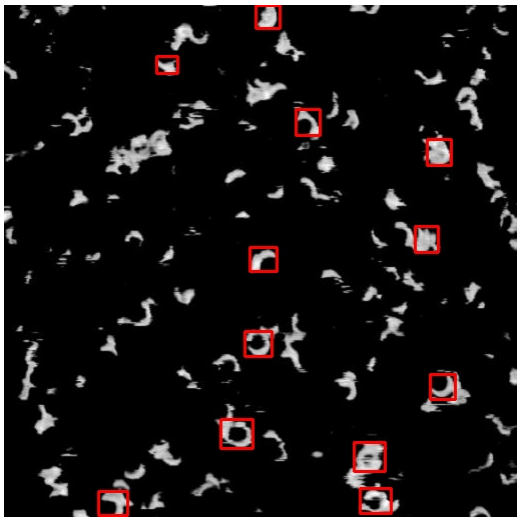
(b) 2 membranos vaizdas.



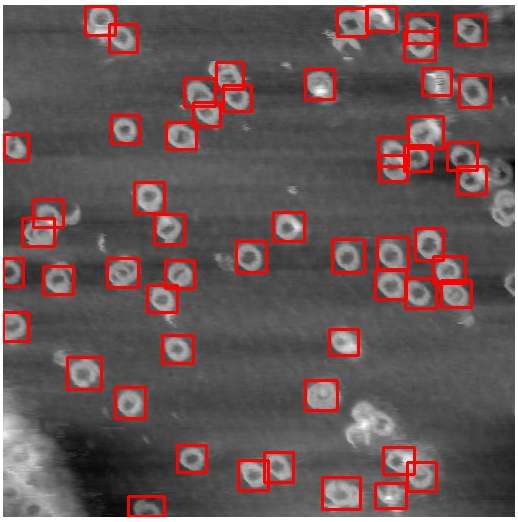
(c) 3 membranos vaizdas.



(d) 4 membranos vaizdas.

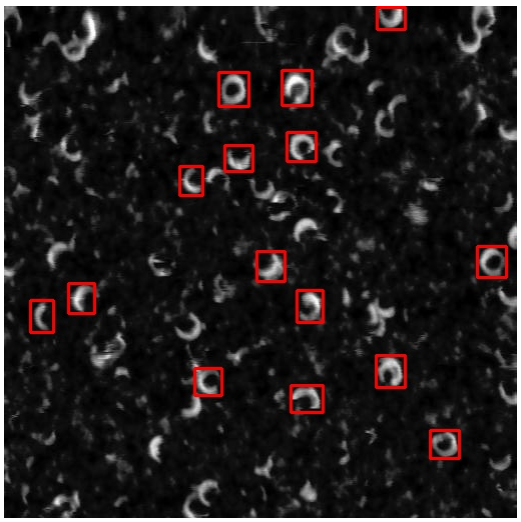


(e) 5 membranos vaizdas.

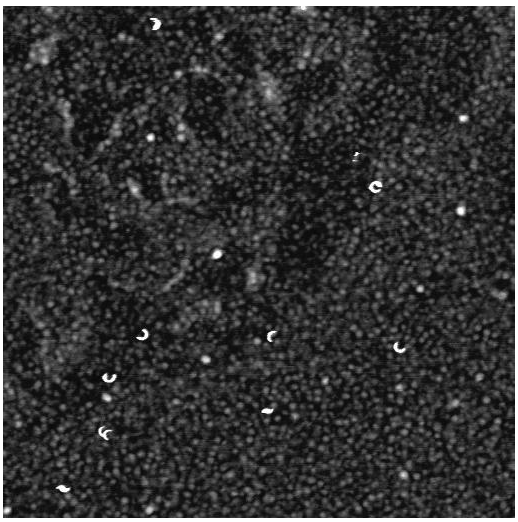


(f) 6 membranos vaizdas.

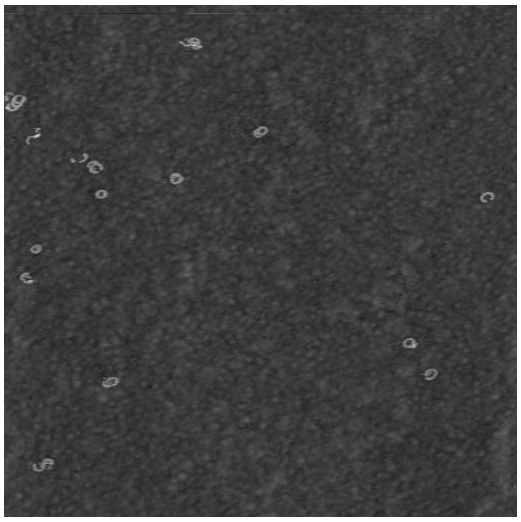
14 pav. EfficientDet D0 512x512 modelio defektų aptikimas testavimo aibės vaizduose.



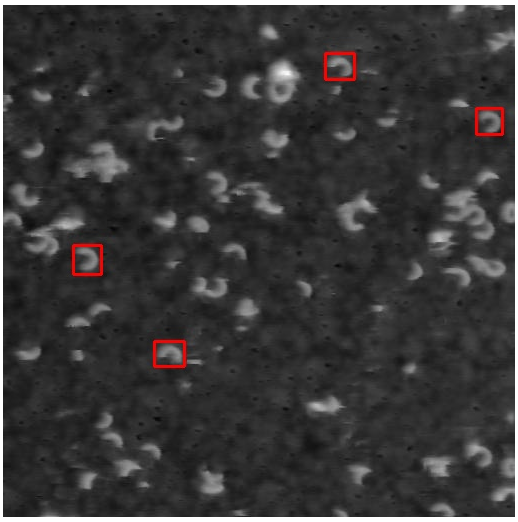
(a) 1 membranos vaizdas.



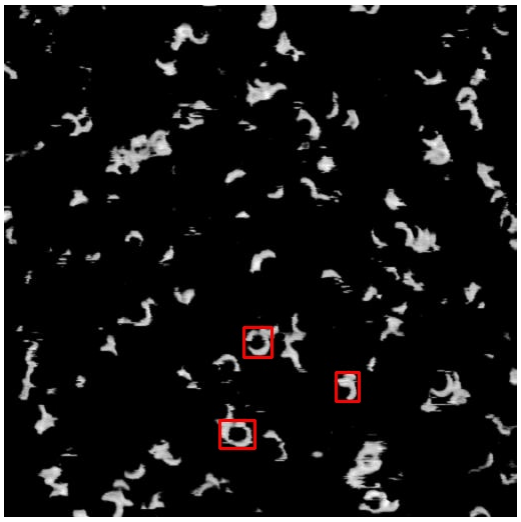
(b) 2 membranos vaizdas.



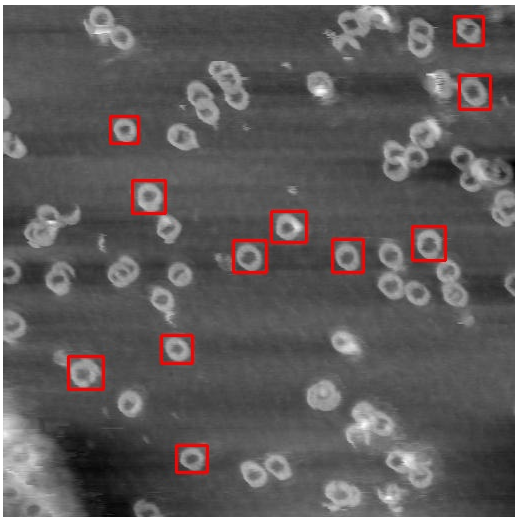
(c) 3 membranos vaizdas.



(d) 4 membranos vaizdas.

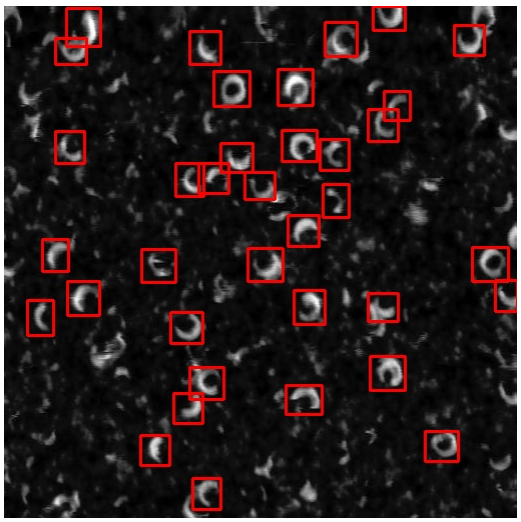


(e) 5 membranos vaizdas.

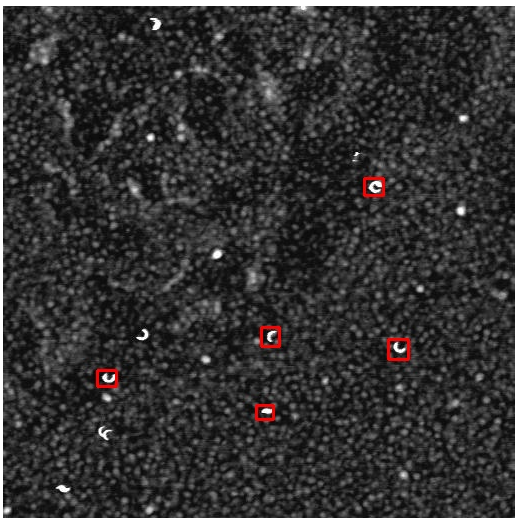


(f) 6 membranos vaizdas.

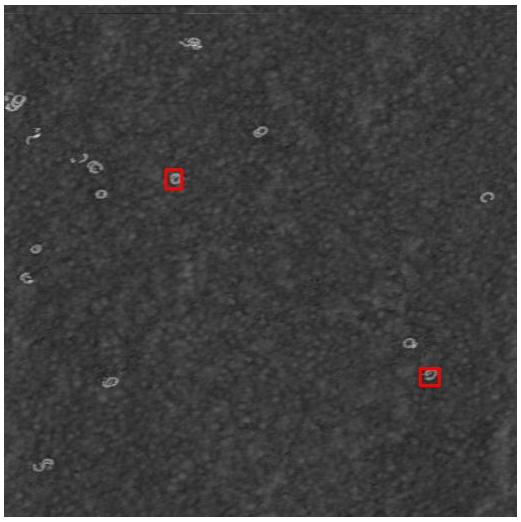
15 pav. SSD ResNet152 V1 FPN 640x640 (RetinaNet152) modelio defektų aptikimas testavimo aibės vaizduose.



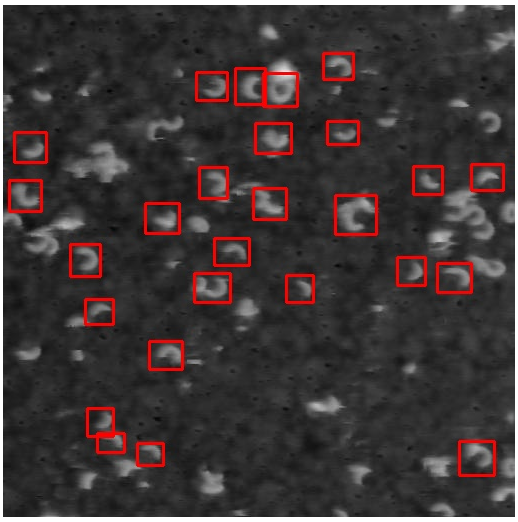
(a) 1 membranos vaizdas.



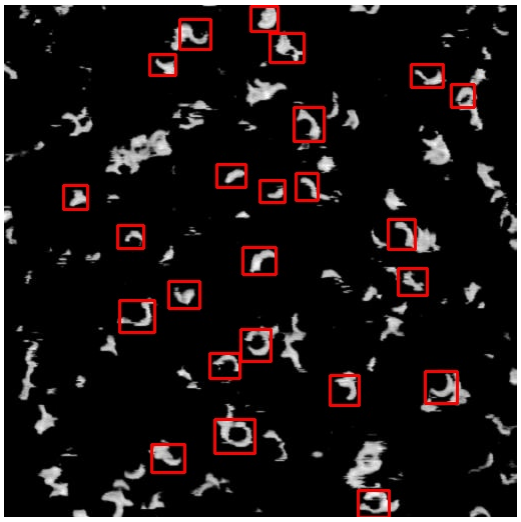
(b) 2 membranos vaizdas.



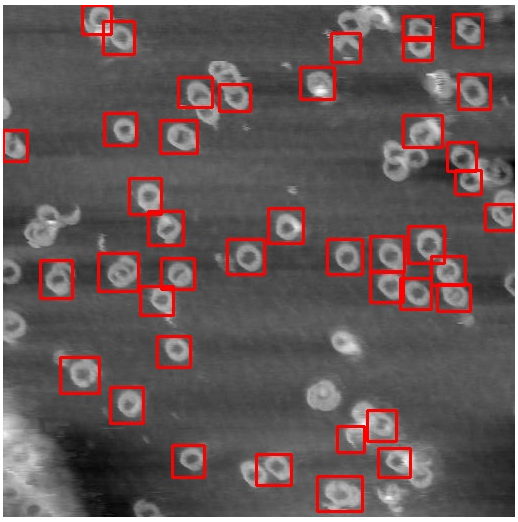
(c) 3 membranos vaizdas.



(d) 4 membranos vaizdas.



(e) 5 membranos vaizdas.



(f) 6 membranos vaizdas.

16 pav. Faster R-CNN Inception ResNet V2 640x640 modelio defektų aptikimas testavimo aibės vaizduose.