Kauno technologijos universitetas

Panevėžio technologijų ir verslo fakultetas

Vaizdo analize pagrįsto optinių indikatorių funkcinių būsenų atpažinimo algoritmo kūrimas ir tyrimas

Baigiamasis magistro studijų projektas

Darius Kapustinskas

Projekto autorius

lektorius-praktikas
dr. Donatas Pelenis

Vadovas

Panevėžys, 2026

Kauno technologijos universitetas

Panevėžio technologijų ir verslo fakultetas

Vaizdo analize pagrįsto optinių indikatorių funkcinių būsenų atpažinimo algoritmo kūrimas ir tyrimas

Baigiamasis magistro studijų projektas

Valdymo technologijos (6211EX014)

Darius Kapustinskas

Projekto autorius

lektorius-praktikas

dr. Donatas Pelenis

Vadovas

Recenzentas (-ė)

Panevėžys, 2026

Kauno technologijos universitetas

Panevėžio technologijų ir verslo fakultetas

Darius Kapustinskas

Vaizdo analize pagrįsto optinių indikatorių funkcinių būsenų atpažinimo algoritmo kūrimas ir tyrimas

Akademinio sąžiningumo deklaracija

Patvirtinu, kad:

1. baigiamąjį projektą parengiau savarankiškai ir sąžiningai, nepažeisdama(s) kitų asmenų autoriaus ar kitų teisių, laikydamasi(s) Lietuvos Respublikos autorių teisių ir gretutinių teisių įstatymo nuostatų, Kauno technologijos universiteto (toliau – Universitetas) intelektinės nuosavybės valdymo ir perdavimo nuostatų bei Universiteto akademinės etikos kodekse nustatytų etikos reikalavimų;

2. baigiamajame projekte visi pateikti duomenys ir tyrimų rezultatai yra teisingi ir gauti teisėtai, nei viena šio baigiamojo projekto dalis nėra plagijuota nuo jokių spausdintinių ar elektroninių šaltinių, visos baigiamojo projekto tekste pateiktos citatos ir nuorodos yra nurodytos literatūros sąraše;

3. įstatymų nenumatytų piniginių sumų už baigiamąjį projektą ar jo dalis niekam nesu mokėjęs(-usi);

4. suprantu, kad išaiškėjus nesąžiningumo ar kitų asmenų teisių pažeidimo faktui, man bus taikomos akademinės nuobaudos pagal Universitete galiojančią tvarką ir būsiu pašalinta(s) iš Universiteto, o baigiamasis projektas gali būti pateiktas Akademinės etikos ir procedūrų kontrolieriaus tarnybai nagrinėjant galimą akademinės etikos pažeidimą.

Darius Kapustinskas
Patvirtinta elektroniniu būdu

Kauno technologijos universitetas

Panevėžio technologijų ir verslo fakultetas

TVIRTINU
TVKC vadovė
Doc. dr. Nida Kvedaraitė

Baigiamojo magistro projekto užduotis

Diplomantui **Dariui Kapustinskui**

Baigiamojo projekto tema (lietuvių kalba)	Vaizdo analize pagrįsto optinių indikatorių funkcinių būsenų atpažinimo algoritmo kūrimas ir tyrimas
Baigiamojo projekto tema (anglų kalba)	Development and Research of an Algorithm for Recognizing the Functional States of Optical Indicators Based on Image Analysis

Patvirtinta 2026 m. balandžio 7 d. dekano potvarkiu Nr. V25-13-9

Parengto baigiamojo projekto įkėlimo į „Moodle“ aplinką terminas iki 2026 m. gegužės 29 d.

Duomenys, reikalavimai ir sąlygos baigiamajam projektui: nėra

Baigiamojo projekto užduotys / uždaviniai, kurie turi būti atskleisti projekte

1. Išanalizuoti elektronikos gaminių optinių elementų testavimo metodus ir nustatyti jų taikymo ribotumus vaizdo analizės pagrindu kuriamai sistemai; 2. Ištirti vaizdų atpažinimo metodus ir pagrįsti tinkamiausio metodo parinkimą segmentų ekranui ir LED būsenoms nustatyti; 3. Suprojektuoti ir eksperimentiškai realizuoti optinių indikatorių testavimo sistemą duomenims rinkti ir algoritmui validuoti; 4. Sukurti vaizdų apdorojimo algoritmą ir programinį sprendimą, skirtą segmentų ekrano ir LED būsenoms nustatyti, įvertinant kintančios geometrijos įtaką tarp kameros ir tiriamojo objekto; 5. Eksperimentiškai ištirti sukurto algoritmo tikslumą, patikimumą ir pritaikomumą realiomis sąlygomis.

Vadovas	lektorius praktikas dr. Donatas Pelenis
	(vadovo pareigos, vardas, pavardė, parašas)
Užduotį gavau	Darius Kapustinskas
	(studento vardas, pavardė, parašas)

2026 m. balandžio 14 d.

Kapustinskas Darius. Vaizdo analize pagrįsto optinių indikatorių funkcinių būsenų atpažinimo algoritmo kūrimas ir tyrimas. Magistro studijų baigiamasis projektas / vadovas lektorius-praktikas dr. Donatas Pelenis; Kauno technologijos universitetas, Panevėžio technologijų ir verslo fakultetas.

Studijų kryptis ir sritis: elektronikos inžinerija, technologijos mokslai.

Reikšminiai žodžiai: vaizdo analizė, YOLOv5, ROI, LED indikatoriai, segmentų ekranas, būsenų atpažinimas, eksperimentinis vertinimas.

Panevėžys, 2026. 104 p.

Santrauka

Baigiamajame magistro projekte tiriamas vaizdo analize pagrįstas optinių indikatorių funkcinių būsenų atpažinimo algoritmas. Projekto aktualumas siejamas su poreikiu elektronikos gaminiuose naudojamus segmentinius ekranus ir LED indikatorius tikrinti greitai, bekontakčiu būdu ir nenaudojant sudėtingos specializuotos įrangos. Tokiam sprendimui įtaką daro objekto padėtis kameros vaizde, apšvietimas, atspindžiai ir ROI sričių pozicionavimo tikslumas.

Projekto tikslas – sukurti ir eksperimentiškai įvertinti algoritmą, skirtą segmentinio ekrano ir LED indikatorių būsenoms atpažinti. Išanalizavus optinių elementų testavimo ir vaizdų atpažinimo metodus nustatyta, kad tikslinga taikyti mišrų sprendimą: YOLOv5 modeliu lokalizuoti tiriamąjį objektą ir segmentinio ekrano sritį, o segmentų ir LED būsenas nustatyti pagal ROI srityse apskaičiuotus šviesumo bei spalvinius požymius.

Eksperimentinę sistemą sudaro segmentų ekranas, du LED indikatoriai, ESP32-S3 mikrovaldiklis, USB kamera ir kompiuteris. Mikrovaldiklis formuoja etalonines būsenas, o kamera fiksuoja jų vaizdus. Sukurtas algoritmas aptinka PCB ir segmentinio ekrano sritį, suvienodina kadro geometriją, sudaro ROI sritis ir pagal jose apskaičiuotus požymius nustato optinių indikatorių būsenas. Programinė dalis realizuota „Python“ aplinkoje, naudojant „OpenCV“, „PyTorch“, YOLOv5 modelį ir UART ryšį su ESP32-S3 valdikliu.

Eksperimentinio tyrimo metu surinktas 6 840 kadrų rinkinys, apimantis skaitines segmentų ekrano reikšmes ir pavienių segmentų būsenas su skirtingomis LED indikatorių kombinacijomis. YOLOv5 lokalizavimo modeliui parengti atrinkti ir sužymėti 75 vaizdai: 60 naudota mokymui, 15 – validacijai. Programos kūrimo metu naudota 15 pavienių segmentų kadrų derinimo imtis, skirta ROI pozicionavimui, slenkstinėms reikšmėms ir sprendimo logikai suderinti.

Vėlesniame tyrimo etape skaitinių reikšmių analizės atsisakyta, nes galutinis vertinimas orientuotas ne į skaičiaus kaip simbolio atpažinimą, o į atskirų optinių elementų „ON“ / „OFF“ būsenų nustatymą. Galutiniam eksperimentiniam vertinimui naudota 840 kadrų imtis, sudaryta iš 14 pavienių segmentų, 4 LED indikatorių kombinacijų ir 15 pakartojimų. Klaidingai atpažinti 7 kadrai, todėl bendras kadrų lygio tikslumas siekė 99,17 proc. Vertinant 14 segmentų „ON“ / „OFF“ būsenas, pasiektas 99,89 proc. tikslumas, o abiejų LED indikatorių būsenos nustatytos 100,00 proc. tikslumu. Klaidų analizė parodė, kad pagrindinės paklaidos atsirado dėl atspindžių, lokalių ryškumo pokyčių ir kameros ekspozicijos prisitaikymo.

Kapustinskas Darius. Development and Research of an Algorithm for Recognizing the Functional States of Optical Indicators Based on Image Analysis. Master's Final Degree Project / supervisor teacher practitioner dr. Donatas Pelenis; Panevėžys Faculty of Technologies and Business, Kaunas University of Technology.

Study field and area: Electronics Engineering, Technology Sciences.

Keywords: image analysis, YOLOv5, ROI, LED indicators, seven-segment display, state recognition, experimental evaluation.

Panevėžys, 2026. 104 pages.

Summary

The Master's final project investigates an image-analysis-based algorithm for recognizing the functional states of optical indicators. The relevance of the work is related to the need to test seven-segment displays and LED indicators used in electronic products quickly, contactlessly, and without complex specialized equipment. The accuracy of such a solution is affected by the position of the object in the camera image, lighting conditions, reflections, and the accuracy of ROI positioning.

The aim of the project is to develop and experimentally evaluate an algorithm for recognizing the states of a seven-segment display and LED indicators. After analyzing optical element testing and image recognition methods, it was determined that a combined approach is suitable for this work: the YOLOv5 model is used to localize the tested object and the display area, while the states of the segments and LEDs are determined using brightness and color features calculated within ROI areas.

The experimental system consists of a seven-segment display, two LED indicators, an ESP32-S3 microcontroller, a USB camera, and a computer. The microcontroller generates reference states, while the camera captures their images. The developed algorithm detects the PCB and seven-segment display area, normalizes the frame geometry, forms ROI areas, and determines the optical indicator states based on the calculated features. The software was implemented in the Python environment using "OpenCV", "PyTorch", the YOLOv5 model, and UART communication with the ESP32-S3 microcontroller.

During the experimental study, an initial dataset of 6,840 frames was collected, including numerical seven-segment display values and individual segment states combined with different LED indicator combinations. To prepare the YOLOv5 localization model, 75 images were selected and annotated: 60 were used for training and 15 for validation. During software development, a tuning set of 15 individual segment frames was used to adjust ROI positioning, threshold values, and decision logic.

At a later stage of the study, the analysis of numerical values was abandoned because the final evaluation focused not on recognizing a displayed number as a symbol, but on determining the individual "ON" / "OFF" states of optical elements. The final experimental evaluation used a dataset of 840 frames, consisting of 14 individual segments, 4 LED indicator combinations, and 15 repetitions. Seven frames were recognized incorrectly, resulting in an overall frame-level recognition accuracy of 99.17 %. When evaluating the "ON" / "OFF" states of 14 segments, an accuracy of 99.89 % was achieved, while both LED indicator states were recognized with 100.00 % accuracy. The error analysis showed that the main inaccuracies were caused by reflections, local brightness variations, and camera exposure adjustment.

Turinys

Paveikslų sąrašas

Lentelių sąrašas

Santrumpų ir terminų sąrašas

Santrumpos:

LED – šviesos diodai (angl. *light emitting diode*);

ROI – dominančios sritys (angl. *region of interest*);

YOLOv5 – penktos versijos atvirojo kodo dirbtinio intelekto modelis, skirtas realaus laiko objektų aptikimui vaizduose ir vaizdo įrašuose (angl. *You Only Look Once* version 5);

PCB – spausdintinė montažinė plokštė (angl. *printed circuit board*).

Įvadas

Aktualumas. Elektronikos gaminiuose plačiai naudojami optiniai indikatoriai, tokie kaip segmentiniai ekranai ir LED indikatoriai. Jų funkcinių būsenų patikra yra svarbi kokybės kontrolės dalis, nes pagal šiuos elementus nustatomas įrenginio veikimo režimas, būsena arba klaidos signalas. Tradiciniai testavimo metodai dažnai reikalauja specializuotos, brangios įrangos arba kontaktinių matavimų, todėl aktualūs tampa bekontakčiai, lengvai pritaikomi vaizdo analizės sprendimai.

Problematika. Segmentinio ekrano ir LED indikatorių būsenų atpažinimą iš kameros vaizdo apsunkina nevienodas apšvietimas, atspindžiai, kintanti padėtis kadre ir ROI sričių pozicionavimo paklaidos. Dėl šių veiksnių vien paprasti slenkstiniai vaizdų apdorojimo metodai ne visada užtikrina stabilų rezultatą. Todėl reikalingas algoritmas, kuris leistų nustatyti tiriamojo objekto padėtį vaizde, suvienodinti kadro geometriją ir pagal apibrėžtas ROI sritis įvertinti optinių indikatorių būsenas.

Tyrimų objektas – vaizdo analize pagrįstas optinių indikatorių funkcinių būsenų atpažinimo algoritmas.

Tikslas – sukurti ir ištirti vaizdo analize pagrįsto optinių indikatorių funkcinių būsenų atpažinimo algoritmą.

Uždaviniai:

1. Išanalizuoti elektronikos gaminių optinių elementų testavimo metodus ir nustatyti jų taikymo ribotumus vaizdo analizės pagrindu kuriamai sistemai;
2. Ištirti vaizdų atpažinimo metodus ir pagrįsti tinkamiausio metodo parinkimą segmentų ekranui ir LED būsenoms nustatyti.
3. Suprojektuoti ir eksperimentiškai realizuoti optinių indikatorių testavimo sistemą duomenims rinkti ir algoritmui validuoti.
4. Sukurti vaizdų apdorojimo algoritmą ir programinį sprendimą, skirtą segmentų ekrano ir LED būsenoms nustatyti, įvertinant kintančios geometrijos įtaką tarp kameros ir tiriamojo objekto.
5. Eksperimentiškai ištirti sukurto algoritmo tikslumą, patikimumą ir pritaikomumą realiomis sąlygomis.

Tyrimų metodai: mokslinių šaltinių analizė, eksperimentinės sistemos projektavimas, duomenų rinkinio formavimas ir anotavimas, YOLOv5 modelio mokymas, ROI sričių formavimas, segmentų ir LED būsenų nustatymo algoritmo kūrimas ir eksperimentinis rezultatų vertinimas.

Konferencijoje(-jose) skaityti pranešimai:

Kapustinskas, Darius; Šviesos diodų segmentų aptikimo vaizdo kamerų sistemomis tyrimas. 25-oji studentų mokslinė konferencija „Technologijų ir verslo aktualijos“. Panevėžys: Kauno technologijos universiteto Panevėžio technologijų ir verslo fakultetas, 2025 m. lapkričio 21 d.

Autoriaus publikuotų straipsnių bibliografinis sąrašas:

Kapustinskas, Darius; Strikulienė, Olga; Pelenis, Donatas. Vaizdo kameros taikymas elektronikos gaminių optinių elementų būsenoms atpažinti // Technologijų ir verslo aktualijos – 2025: studentų mokslinės konferencijos pranešimų medžiaga, Lietuva, Panevėžys, 2025 m. lapkričio 21 d., 455–464 psl. / Kauno technologijos universiteto Panevėžio technologijų ir verslo fakultetas. Kaunas: Kauno technologijos universitetas. ISSN 2538-8045. 2025.

1. Elektronikos gaminių optinių elementų testavimo metodai ir algoritmai

Šiame skyriuje teorinė optinių elementų testavimo metodų analizė apribojama konkrečiu šio darbo objektu – segmentinio ekrano ir dviejų papildomų LED indikatorių funkcinių būsenų atpažinimu iš kameros vaizdo. Todėl literatūroje nagrinėjami metodai vertinami ne pagal jų tinkamumą absoliučiai matuoti spektrinius, elektrinius ar šiluminius optinių elementų parametrus, o pagal tai, ar jie gali būti pritaikyti automatizuotai nustatyti, ar konkretus indikatorius yra įjungtas, ar išjungtas.

Alternatyvūs kontaktiniai, spektriniai, talpiniai ir kiti specializuoti optinių elementų testavimo metodai šiame darbe nagrinėjami kaip pramoniniai sprendimai, su kuriais lyginama pasirinkta vaizdo analizės kryptis. Tokie metodai yra reikšmingi tiksliai komponentų diagnostikai, tačiau šio darbo uždaviniui jie dažnai yra pertekliniai, nes čia nereikia nustatyti absoliučių optinių parametrų – reikia patikimai identifikuoti funkcines „ON“ / „OFF“ būsenas.

Dėl to daugiausia dėmesio skiriama vaizdo kamerų taikymui, automatinei optinei inspekcijai, giluminio mokymosi modeliais pagrįstam objektų aptikimui, ypač YOLO tipo algoritmams, kadro geometrijos suvienodinimui ir ROI pagrindu veikiančiai būsenų nustatymo logikai. Tokia analizės kryptis tiesiogiai pagrindžia šiame darbe pasirinktą sprendimą: PCB ir segmentinio ekrano sritis aptikti YOLO modeliu, o segmentų ir LED būsenas nustatyti pagal iš aptiktos geometrijos suformuotas ROI sritis.

1.1. Klasikiniai ir patentiniai optinių elementų testavimo metodai

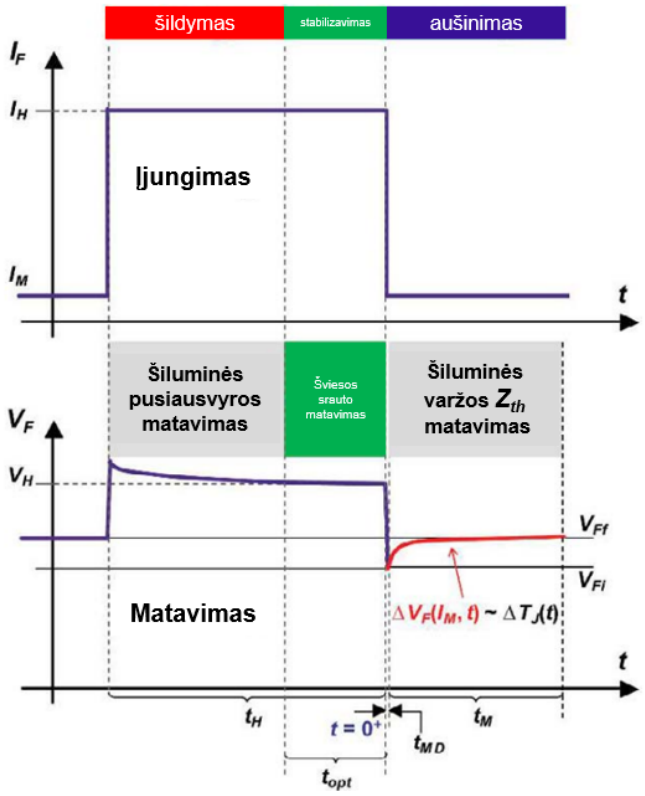
Klasikiniai ir patentiniai optinių elementų testavimo metodai šiame poskyryje lyginami pagal penkis kriterijus: matavimo tikslumą, įrangos sudėtingumą ir kainą, bandymo trukmę, kontaktinį arba bekontaktį veikimo principą ir tinkamumą segmentų bei LED „ON“ / „OFF“ būsenoms nustatyti. Klasikiniai metodai dažniausiai remiasi fiziniais parametrais – šviesos intensyvumu, spektriniu pasiskirstymu, atspindžio charakteristikomis ir mechanine sąveika. Jie tinkami tada, kai reikia tiksliai išmatuoti komponento savybes, tačiau šiame darbe svarbiausia ne absoliutus optinis parametras, o patikimas funkcinės būsenos nustatymas iš kameros kadro.

Optinių LED elementų testavimas tradiciškai vykdomas izoterminių IVL (srovės-įtampos-šviesos srauto, angl. *current-voltage-luminance*) charakteristikų matavimu – nuosekliai keičiant šiluminį ir elektrinį apkrovimą, užfiksuojant IVL priklausomybes (žr. 1 pav.) [1]. Pagal standartizuotas laboratorines procedūras, atitinkančias JEDEC (jungtinė elektroninių įtaisų inžinierių taryba, angl. *joint electron device engineering council*) ir CIE (tarptautinė apšvietimo komisija, angl. *international commission on illumination*) rekomendacijas, reikia stabilizuoti jungties temperatūrą ir srovę kiekviename taške, o testavimo ciklas užtrunka nuo 2 iki 5 min. vienam LED, kai naudojama automatizuota sistema (pvz.: „Siemens Simcenter Micred T3Ster“ su „TeraLED“ ir CAS140 spektrometru) [1]. Tipiniai srovės intervalai – 10–1000 mA, temperatūrų intervalai – 30–110 °C. Pilnam LED tipo matavimų ciklui (33 srovės / temperatūros kombinacijos, kiekvienam matuojamam 5 LED rinkiniui) reikia 6–8 val., o tai lemia ribotą našumą masinėje gamyboje [1]. Dėl to vystomi spartesni testavimo metodai, leidžiantys sumažinti bendrą testavimo trukmę apie 10 kartų, testuojant 9–25 LED rinkinius.

Vienas iš tokių metodų – lygiagretus kelių LED rinkinių testavimas vienu metu, naudojant trumpųjų impulsų optinius matavimus pagal CIE 226:2017 metodiką. Šiuo metodu optinės savybės matuojamos impulsiniu režimu, o šiluminiai perėjimai registruojami lygiagrečiai keliems LED, tai

leidžia esminę proceso dalį – temperatūros keitimą – atlikti greičiau [1]. Duomenų koregavimas vykdomas pagal CIE 225:2017 rekomendacijas, jis leidžia tiksliau apskaičiuoti momentinę jungties temperatūrą [1].

Tarp klasikinių eksperimentinių tyrimų išsiskiria kontaktinės jėgos matavimo metodas mikro-LED panelėms [2]. Čia sukurtas optinio sverto principu pagrįstas matavimo metodas: naudojamas mikro-jautrus svirtelės jutiklis, lazerio šaltinis ir jautrus padėties detektorius. Atliekamas optinis kalibravimas ir eksperimentinis jėgos matavimas: nustatyta, kad patikimas vieno LED elemento ir montažinės plokštės sujungimo stipris turi būti ne mažesnis nei 2,58 mN. Eksperimentas parodė, kad vien tik padėties nuokrypio analizė nėra pakankama, būtina taikyti kiekybinį sukibimo jėgos matavimą, tai leidžia optimizuoti procesų parametrus ir identifikuoti defektus gamybos eigoje [2].



1 pav. LED testavimo rezultatų grafikų rekomendacijos, pagal JESD51 standartą [1]

Patentiniuose šaltiniuose (ypač taikomuose masinėje gamyboje) akcentuojama kontaktinių zondų eliminacija, siekiant išvengti kontaktų pažeidimų ir užtikrinti didesnį testavimo našumą. Pavyzdžiui, „Apple“ patentuotas metodas [3] leidžia funkcionaliai testuoti visą LED masyvą be tiesioginio elektrinio kontakto – įtampa, bangos forma paleidžiama per izoliacinę plokštę (angl. *field plate*), taip suformuojant talpinį krūvį. Taip į kiekvieną LED tuo pačiu metu paleidžiamas srovės impulsas, o šviesos emisija fiksuojama kamera. Keičiant įtampos režimus galima matuoti, pavyzdžiui, išorinį kvantinį efektyvumą arba identifikuoti defektinius LED (naudojant spektrinius filtrus ir papildomą apšvietimą matavimų kontrastui didinti). Metodo privalumas – pritaikant kelių kamerų masyvus ar naudojant žingsnio kartojimo (angl. *step / repeat*) principą, galima vienu metu testuoti milijonus LED. Taip eliminuojamas poreikis šimtams ar tūkstančiams kontaktinių zondų, išvengiama kontaktinių defektų, didinamas testavimo patikimumas ir sumažinamos sąnaudos [3].

Kiti patentuoti sprendimai [4–5] telkia dėmesį į optinius, spektrinius ir filtravimo metodus: siūlomas matavimo principas, kai LED plokštelė apšviečiama šviesos šaltiniu, pro jį pereina spektras, kuris vėliau filtruojamas specialiais filtrais ir fiksuojamas atitinkamoje spektro srityje. Tai leidžia tiksliai nustatyti, pavyzdžiui, pralaidumą, spalvinį nuokrypį ar aptikti sugedusius plotus visoje plokštelėje vienu žingsniu. Taip realizuojama LED masyvo kokybės kontrolė masinės gamybos sąlygomis, sumažinant analizės laiką ir žmogiškųjų klaidų riziką.

Patentuota mikro-LED testavimo technologija [6] aprašo lanksčių zondų naudojimą defektų diagnostikai, tai leidžia atsekti net ir labai mažų matmenų lustų defektus tiek gamybos, tiek remonto stadijoje.

Tarp patentinių sprendimų reikšminga vieta skiriama ir sugedusių LED elementų remonto metodams serijiniuose ekranuose [7]. Autorių pateiktas patentas aprašo sistemą ir metodiką, skirtą serijiniu būdu sujungtų ekrano elementų (tokie naudojami LED matricose) remontui: sugedęs elementas gali būti apeinamas ar pakeičiamas specialiu elektroniniu perjungimu ar integruotomis remonto grandinėmis. Tai ypač aktualu didelio tankio LED ekranuose, kur net pavienio sugedusio elemento netaisymas gali lemti visos matricos veikimo sutrikimą ar ryškų vaizdo kokybės sumažėjimą. Sistemos architektūra leidžia atlikti tiek automatinį defekto nustatymą, tiek realiu laiku taikyti remonto algoritmus, o tai sumažina nereikalingų modulių keitimo ar masinės gamybos praradimus ir padidina galutinio produkto patikimumą [7]. Šis metodas papildo kontaktinius, optinius ir talpinius testavimo bei diagnostikos principus, nes ne tik leidžia greitai (palyginti su visa tikrinimo trukme) aptikti defektą, bet ir taikyti tikslingą remonto sprendimą be būtinybės pakeisti visą ekraną ar modulį. Taip mažinamos remonto sąnaudos, trumpinamas prastovų laikas ir užtikrinamas didesnis LED ekranų tarnavimo laikas bei kokybė masinėje gamyboje [7].

Moksliniame straipsnyje [8] pasiūlyta ir bekontaktė (angl. *contactless*) mikro-LED diagnostika per fotovoltinį efektą – matuojamas įtampų skirtumas tarp anodų ir katodų, naudojant talpinį vaizdo jutiklį, kai LED apšviečiami mažesnės nei jų spinduliuojamos ilgio bangos šviesa. Tai leidžia vienu metu be žalos įvertinti daugiau nei 50 000 mikro-LED (60 × 34 μm) per ~2 s. Šiuo metodu galima aptikti tiek atvirų, tiek trumpų jungimų gedimus, nes atviram sujungimui fotovoltinė įtampa VOC neatsiranda, o trumpinant – lygi nuliui. Metodas eliminuoja poreikį kontaktiniams zondams, spartina masinę diagnostiką ir sumažina defektų perėjimo į surinktą gaminį riziką.

Bekontakčiai patentiniai ir moksliniai sprendimai yra artimesni šio darbo krypčiai, nes juose taip pat siekiama mažinti kontaktinių zondų poreikį ir naudoti optinį fiksavimą. Vis dėlto dauguma jų orientuoti į LED masyvų, „mini-LED“ arba mikro-LED gamybinę diagnostiką, kur reikalinga specializuota įranga, filtrai, talpiniai jutikliai arba valdomas sužadinimas. Šiame projekte pasirenkamas paprastesnis sprendimas – kamera, objekto lokalizavimas vaizde ir ROI analizė, nes tiriamasis objektas turi aiškiai apibrėžtas indikatorines būsenas.

Apibendrinant galima teigti, kad klasikiniai LED optinių elementų testavimo metodai dėl būtinybės stabilizuoti parametrus pasižymi lėta testavimo eiga, tuo tarpu naujausi patentiniai ir moksliniai metodai diegia lygiagretų testavimą, talpinį ir optinį sužadinimą bei bekontakčius matavimus, tai leidžia sumažinti testavimo laiką nuo dienos iki kelių valandų ar minučių, išvengiant kontaktinių defektų ir užtikrinant didesnį našumą masinėje gamyboje.

Šių metodų analizė parodo, kad tikslūs optinių parametrų matavimo sprendimai dažniausiai reikalauja specializuotos įrangos, stabilizuotų bandymo sąlygų, spektrinių matavimo priemonių arba sudėtingų

kontaktinių ir bekontakčių sužadinimo schemų. Toks sprendimas tinkamas pramoninei LED ar mikro-LED masyvų patikrai, tačiau šiame projekte sprendžiamas siauresnis uždavinys – nustatyti segmentinio ekrano ir dviejų LED funkcines būsenas, t. y. ar konkretus optinis indikatorius yra įjungtas, ar išjungtas. Todėl tolesniam tyrimui pasirinkta paprastesnė ir pigesnė vaizdo kameros kryptis, kai tiriamojo objekto padėtis nustatoma vaizde, geometrija suvienodinama, o segmentų ir LED būsenos vertinamos algoritmiškai pagal ROI sritis.

1.2. Automatizuotų kokybės kontrolės metodų raida ir taikymas pramonėje

Šiuolaikinėje elektronikos ir optoelektronikos pramonėje automatizuota kokybės kontrolė ir defektų aptikimas tapo esminiais procesais norint išlaikyti aukštą gaminių kokybę ir sumažinti žmogiškų klaidų tikimybę. Mokslinės literatūros [9–19] apžvalga rodo aiškią metodų evoliuciją – nuo klasikinių vaizdų apdorojimo algoritmų iki pažangių giluminio mokymosi modelių, kurie geba automatiškai mokytis iš didelių duomenų kiekių ir prisitaikyti prie naujų gaminių ar defektų tipų.

Tyrimai [9–12] pabrėžia, kad tradiciniai vaizdų apdorojimo metodai, tokie kaip šablonų atitikimas, ribų aptikimas ar morfologinė analizė, tebėra naudojami PCB (spausdintinė montažinė plokštė, angl. *printed circuit board*) ir kitų optinių elementų defektų aptikime, tačiau jų veiksmingumas dažnai priklauso nuo apšvietimo sąlygų, komponentų variacijų ir reikalauja rankinio parametrų derinimo bei ekspertinių žinių. Šie metodai gali būti efektyvūs, kai defektai pasižymi aiškiais geometriniais bruožais, tačiau dažnai yra riboti sprendžiant sudėtingesnes arba nestandartines užduotis, pvz., aptinkant subtilius ar naujus defektų tipus.

Šių metodų grupė šiam darbui svarbi kaip atskaitos taškas: klasikinis vaizdų apdorojimas yra greitas ir aiškiai interpretuojamas, bet nepakankamai patikimas, kai keičiasi apšvietimas, ekspozicija ir PCB padėtis. Todėl vien slenkstinis arba šabloninis sprendimas pasirinktam objektui negali būti pagrindinis metodas.

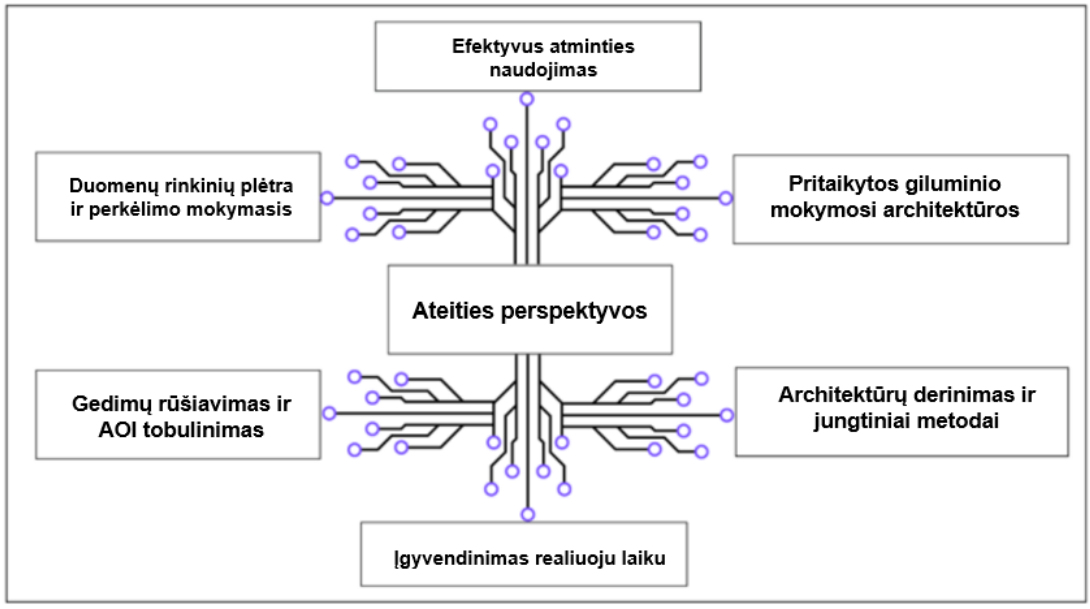
Giluminio mokymosi modeliai, ypač konvoliuciniai neuroniniai tinklai (angl. *convolution neural network*, CNN), pastaraisiais metais tapo pagrindiniu instrumentu tiek optinių, tiek elektronikos gaminių kokybei kontroliuoti. CNN naudojami ne tik vaizdams klasifikuoti, bet ir defektams klasifikuoti, anomalijoms aptikti, netiesioginiams parametrams prognozuoti [9, 11, 13–18]. Pvz., mokslininkas [13] savo disertacijoje detaliai aprašė WM-811K puslaidininkių „plokštelių dėžės žemėlapio“ (angl. *wafer bin map*) duomenų rinkinio analizę, kur tradiciniai metodai buvo papildyti DCGAN (angl. *deep convolutional generative adversarial network*) – giliuoju konvoliuciniu generatyviniu priešpriešiniu tinklu duomenims praplėsti ir pasiūlytas kapsulinių tinklų (angl. *capsule network*) modelis „WaferCaps“ (specialiai puslaidininkių plokštelių defektams aptikti sukurtas kapsulinių neuroninių tinklų modelis). Šio darbo eksperimentiniai rezultatai parodė, kad pritaikius DCGAN dirbtinei gedimų duomenų generacijai, mokymo tikslumas siekė 99,59 proc., patikrinimo – 97,53 proc., o testavimo – 91,4 proc. Analizuojant optoelektroninių plokštelių gedimų klasifikavimą, CNN ir „WaferCaps“ modelių sprendimų sujungimas padidino bendrą tikslumą iki 98,5 proc.

PCB defektų aptikimo srityje daugelis tyrimų apžvelgia klasikinio vaizdų apdorojimo ir mašininio mokymosi derinius, tačiau vis daugiau darbų remiasi giliojo mokymosi architektūromis (žr. 2 pav.), kurios nereikalauja rankinio požymių išrinkimo ir adaptuojasi prie naujų duomenų [11–12, 15–16]. Autoriai [12] apžvelgė daugiau nei 100 publikacijų, išskirdami, kad giluminio mokymosi modeliai, ypač CNN, LSTM (ilgalaikės trumposios atminties tinklas, angl. *long short-term memory*) ar auto enkoderiai, demonstruoja aukštesnį aptikimo tikslumą palyginti su klasikine segmentacija ar šablonų

palyginimu. Viename eksperimentų [11] buvo lyginamas SVM (palaikančiųjų vektorių mašina, angl. *support vector machine*) ir CNN veikimas – CNN pasiekė 99,1 proc. tikslumą, kai tuo tarpu SVM – 95,3 proc., naudojant tą patį PCB duomenų rinkinį.

Naujausi apžvalginiai straipsniai [9, 15–16] taip pat išskiria giluminio mokymosi modelių gebėjimą automatiškai generuoti požymius, o tai yra ypač svarbu didelio sudėtingumo gaminiuose ar nestandartiniuose defektų scenarijuose. Tyrėjai [15] apžvelgia, kad giluminio mokymosi pagrindu veikiančios sistemos leidžia aptikti net subtilius PCB defektus, kurių nepastebi klasikinės metodikos, ir teigia, jog daugelyje testų CNN pagrindu veikiančių metodų tikslumas viršijo 98 proc. Savo ruožtu, mokslininkai [16] akcentuoja, kad AOI (automatinė optinė patikra, angl. *automatic optical inspection*) pagrindu veikiančios sistemos iš esmės keičia kokybės kontrolės praktiką gamyboje dėl jų pritaikomumo, greičio ir mažesnių veiklos kaštų palyginti su rentgeno ar ultragarso metodais.

Kiti tyrimai rodo [14, 17–18], kad dirbtinio intelekto pagrindu veikiančios kokybės kontrolės sistemos pasižymi lankstumu ir gebėjimu apdoroti didelius duomenų kiekius realiu laiku, kas ypač svarbu masinės gamybos kontekste. Autoriai [14] analizavo įvairių giluminio mokymosi ir kompiuterinės regos algoritmų taikymą pramoninėse linijose, pažymėdami, kad CNN pagrindu veikiančios sistemos, palyginti su klasikinėmis, sugeba aptikti daugiau defektų su mažiau klaidingų pozityvių signalų, t. y. CNN pagrindu veikiančios sistemos aptiko iki 99 proc. visų defektų, kai klasikiniai metodai pasiekė tik 92 proc., o klaidingų pozityvių signalų sumažėjo nuo 8 proc. (klasikiniai) iki 2 proc. (CNN).



2 pav. Ateities perspektyva PCB defektų aptikimo srityje [11]

Mokslininkai [18] taip pat pabrėžia, kad AI pagrindu veikiančios sistemos leidžia užtikrinti aukštesnį inspektavimo greitį (AI pagrindu veikiančios sistemos tikrinimo greitį padidino nuo 150 iki 500 vienetų per minutę, kai įprastiniais metodais patikrinama 80–120 vienetų per minutę), sumažinti žmogiškojo faktoriaus įtaką ir generuoti kokybės analizės duomenis tolesniam procesų optimizavimui. Tačiau ir šie, ir kiti šaltiniai atkreipia dėmesį į būdingus iššūkius: didelių žymėtų duomenų rinkinių poreikį, integracijos su esamomis sistemomis sudėtingumą bei poreikį užtikrinti duomenų kokybę ir modelių patikimumą ilgalaikėje perspektyvoje.

Tyrimuose apie LED, ekranų ar kitų optinių komponentų testavimą, kompiuterinės regos sistemos dažniausiai remiasi „OpenCV“ (atviroji kompiuterinė rega, angl. *open computer vision*) pagrindu sukurtais klasikiniais algoritmais, tačiau naujesni tyrimai įtraukia giluminio mokymosi sprendimus, kurie lengviau adaptuojami skirtingoms testavimo užduotims, sumažina klaidingų signalų kiekį ir geriau atpažįsta silpnai matomus defektus [10, 17]. Pavyzdžiui, tyrėjas [19] savo darbe pateikė kompiuterinės regos sprendimo taikymą LED panelių testavimui, kuriame naudojant „OpenCV“ pagrindu sukurtą programos kodą, 64 × 32 LED panelėje buvo aptikti neveikiantys šviesos diodai – atliktas eksperimentas parodė, kad sistema atitiko visus funkcionalumo ir patikimumo reikalavimus.

Analizuoti DI ir kompiuterinės regos sprendimai rodo, kad giluminis mokymasis ypač naudingas objekto ar defekto lokalizavimo užduotyse. Tačiau šio darbo atveju galutinis sprendimas turi būti susietas su konkrečiais fiziniais segmentais ir LED indikatoriais, todėl vien bendras vaizdo klasifikatorius būtų mažiau aiškus ir sunkiau pritaikomas keičiant ROI išdėstymą.

1.3. Optinių elementų (LED, mini-LED, mikro-LED) testavimo taikomieji ir algoritminiai sprendimai

Šiuolaikinė optinių elementų, tokių kaip LED, mini-LED ir „mikro-LED“, testavimo sritis pasižymi itin plačiu tiek taikomųjų, tiek algoritminių sprendimų spektru, apimančiu vaizdo atpažinimo, spektrinės analizės, giluminio mokymosi ir hibridinių metodų taikymą. Šioje apžvalgoje aptariamos šiuo metu mokslinėje literatūroje aprašytos pagrindinės technologijos, eksperimentai ir pasiekti rezultatai.

Vienas iš reikšmingų LED testavimo metodų – SPD (spektrinės galios skirstinio, angl. *spectral power distribution*) analizė, leidžianti ne tik įvertinti šviesos šaltinio šviesos intensyvumo bei spalvinius parametrus, bet ir atlikti ankstyvą gedimų diagnostiką. LED testavimo metu, pasitelkus SPD modelį ir mašininio mokymosi algoritmus PCA (pagrindinių komponentų analizė, angl. *principal component analysis*), KNN (*k* artimiausių kaimynų algoritmas, angl. *k-Nearest Neighbors*), buvo parodyta, kad anomaliniai šviesos šaltinių pokyčiai gali būti aptikti 789,6 val. nuo eksploatacijos pradžios, palyginti su 1311 val., kol apšvietimo degradacija pasiekia 70 proc., kaip reikalaujama pagal IES TM-21 standartą [20]. Tokie rezultatai rodo mašininio mokymosi metodų pranašumą laiko ir tikslumo prasme prieš tradicinius gedimų nustatymo procesus.

Mini-LED ir mikro-LED technologijose ypatingas dėmesys skiriamas optinei partijų patikrai. Tam taikomas mikroskopinis hiperspektrinis vaizdavimas, leidžiantis vienu metu nustatyti tikslius kiekvieno LED elemento spektrinius parametrus ir spalvų koordinates. Optinė partijų patikros metodika, paremta optimizuotu „Canny“ (kraštų aptikimo) algoritmu, užtikrino efektyvų emituojančių zonų nustatymą dideliuose LED masyvuose, o gautų rezultatų nuokrypis, palyginti su tradiciniu sferiniu integruojančiu metodu, neviršijo 3,2 proc. [21]. Šio metodo eksperimentuose su mini-LED (200 × 100 μm) ir mikro-LED (10 × 10 μm) buvo pademonstruota, kad galima įvertinti kvantinio efektyvumo ir chromatiškumo koordinates, be to, buvo kiekybiškai įvertinta optinė kryžminė įtaka tarp spalvų kanalų – ji ryškiausia pasirodė raudonų LED atveju.

Didelę reikšmę turi ir automatizuota be žymėtų duomenų defektų paieška. Anomalijų detekcija, paremta auto enkoderių architektūra ir rezidualinių (su pridedama liekana) tinklų taikymu, leido mikro-LED atveju pasiekti 95,82 proc. AUROC (kiekybinis modelio gebėjimo atskirti klases įvertinimas, angl. *area under receiver operating characteristic curve*) rezultatą, t. y. 0,67 proc. aukštesnį nei „Deep SVDD“ (giluminio mokymosi metodas anomalijų detekcijai, angl. *support*

vector data description) bei 20,87 proc. aukštesnį nei įprastas konvoliucinis auto enkoderis [22]. Toks metodas eliminuoja klasės disbalanso problemas, kurios būdingos pramoninėje gamyboje, kai didžioji dalis produktų yra geri, o defektų – labai mažai.

Giliųjų neuroninių tinklų taikymas užtikrina itin didelį defektų atpažinimo tikslumą. DENC-CNN (dvigubos entropijos kontrolės konvoliucinis neuroninis tinklas, angl. *dual entropy-controlled CNN*) algoritmas, įvedęs savitas entropijos kontrolės funkcijas, padidino mini- / mikro-LED defektų atpažinimo tikslumą iki 99,12 proc., geometrinis vidurkis tarp jautrumo ir specifiškumo („*G*-mean“) siekė 97,86 proc., rodiklis, apibūdinantis modelio tikslumo ir atšaukimo pusiausvyrą („*F1*-measurement“) – 97,87 proc. Eksperimentuose buvo naudojama specialiai surinkta mini- / mikro-LED vaizdų duomenų bazė [23]. Analogiškai, MDCA-DETR (defektų aptikimo modelis, pagrįstas transformerių architektūra su papildomomis deformuojamosiomis konvoliucijomis ir koordinatiniu dėmesiu, angl. *multi-channel deformable convolution with coordinate attention – detection transformer*), papildytos deformuojamosiomis konvoliucijomis ir koordinatine dėmesio moduliacija, laboratorijoje pasiekė mAP50 (vidutinis tikslumas, kai aptikimas laikomas teisingu, jei numatyta defekto vieta sutampa su tikra bent 50 proc.) reikšmę 90,4 proc., bei mAP50:95 (vidutinis tikslumas per įvairius sutapimo slenksčius nuo 50 iki 95 proc.) – 62,8 proc., o aptikimo sparta viršijo 30 kadrų per sekundę, kai gamyboje reikalaujama, kad 8 colių mini-LED plokštės klaidingo neaptikimo rodiklis būtų mažesnis nei 0,05 proc., klaidingo aptikimo rodiklis – mažesnis nei 1 proc., o aptikimo trukmė – iki 80 s [24].

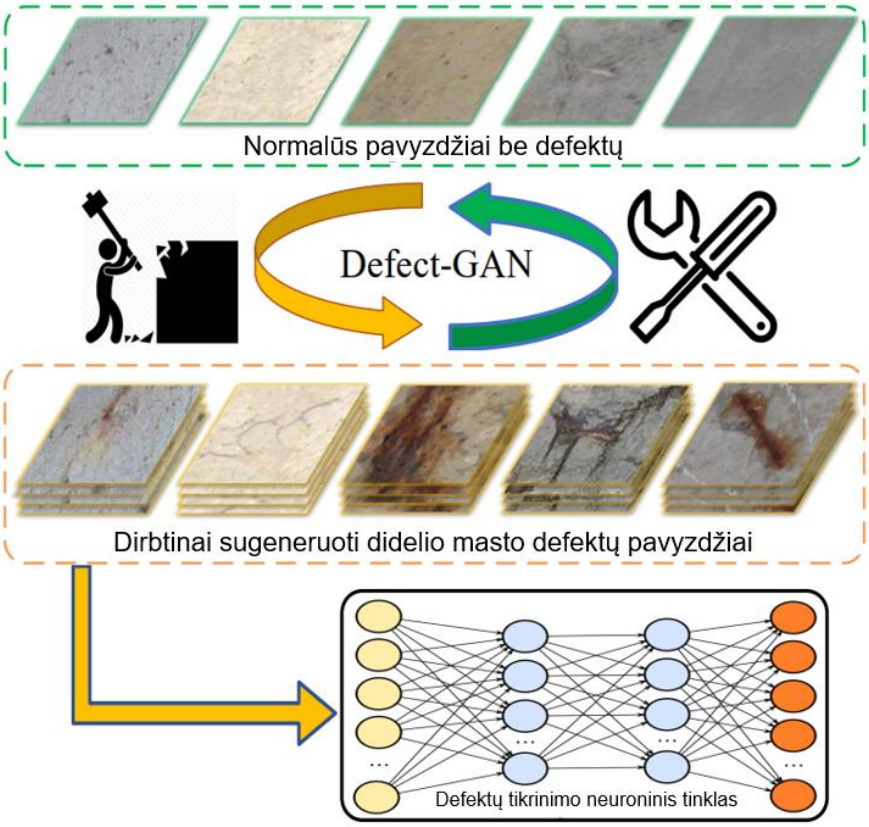
Viename iš naujausių tyrimų [25] giluminio mokymosi metodai buvo pritaikyti LED mikroschemų kokybės saugumui stebėti, integruojant CNN į automatinės optinės inspekcijos (AOI) sistemą. Modelis buvo apmokytas realiais vaizdais paimtais iš gamybos ir leido atpažinti įvairių tipų defektus, įskaitant ir mažai išreikštus pažeidimus, kurie dažnai lieka nepastebėti tradicinėmis vizualinės inspekcijos priemonėmis. Eksperimentų rezultatai parodė, kad pasiūlytas sprendimas pasiekė 99,1 proc. aptikimo tikslumą ir užtikrino reikiamą analizės spartą realaus laiko gamybos procesuose. Palyginti su tradicinėmis metodikomis, šis sprendimas pasižymėjo didesniu universalumu skirtingų partijų ir skirtingų apšvietimo sąlygų atveju, o sistema leido sumažinti klaidingų neaptikimų kiekį iki 0,9 proc. ir užtikrinti aukštesnę LED gaminių kokybę [25].

Aptikimo tikslumą dar labiau didina „dėmesio sau“ (angl. *self-attention*) mechanizmais pagrįsti sprendimai, tokie kaip CM-YOLOv5 (modifikuotas dirbtinio intelekto algoritmas, paremtas YOLO (angl. *you only look once*) objektų aptikimo modeliu), kuris vienu metu analizuoja visą vaizdą ir automatiškai aptinka defektus, o papildyti „dėmesio sau“ mechanizmai leidžia tiksliau atpažinti net ir labai mažus LED defektus. Ši sistema naudojama mikro-LED defektams lokalizuoti ir klasifikuoti, naudoja „kelių kanalų dėmesio sau mechanizmą“ (angl. *multi-head self-attention*), SIoU (sutapimo įvertinimo metodas, angl. *scylla-intersection over union*) bei klasių disbalansui pritaikytą nuostolių funkciją (angl. *class-balanced BCE*). Eksperimentuose siūlomas sprendimas palygintas su baziniu YOLOv5 algoritmu. Gauti rezultatai: mAP (vidutinis tikslumas, angl. *mean average precision*) padidėjo 3,8 proc., tikslumas – 3,7 proc., o įdiegus sistemą „NVIDIA Jetson Xavier NX“ įrenginyje, buvo pasiekti įterptinėse sistemose aktualūs aptikimo spartos (39,5 kadro per sekundę) ir tikslumo (99,1 proc.) rodikliai [26].

Kitoje publikacijoje [18] akcentuojama įvairių hibridinių ar segmentavimo metodų taikymo nauda. Efektyvių segmentavimo tinklų, tokių kaip „Hrnet“ (tinklas, kuris nuolat išlaiko aukštą vaizdo raišką per visą apdorojimą, angl. *high-resolution network*) su globalaus konteksto likutinio dėmesio

mechanizmu (RGCA), MDC-DUC moduliu, naudojimas leido pasiekti „mIoU“ (vidutinio sutapimo su tikrąja defekto sritimi rodiklis, angl. *mean intersection over union*) 86,91 proc. Mini-LED apšvietimo panelių inspekcijoje buvo pasiektas mažesnis vidutinės klaidos rodiklis (RMSE), palyginti su kitais giluminio mokymosi segmentavimo modeliais, tokiais kaip „Unet“ (giluminio mokymosi segmentavimo modelis), „PSPNet“ (segmentavimo tinklas), „Deeplabv3+“ (modernus giluminio mokymosi segmentavimo modelis) ar įprastinis „Hrnet“, tai reiškia, kad defektų vietos buvo pažymėtos tiksliau nei naudojant minėtus metodus. Kitame darbe [27] naudojant STAE-Net tinklą (angl. *student-teacher autoencoder*), pasiekta 99,1 proc. defektuotų ir gerų vaizdų atskyrimo viso vaizdo mastu ir 98,3 proc. atpažinimo kiekvieno taško tikslumu, ar pikselis priklauso defektui, ar ne, o apdorojimo sparta siekė 125 kadrų per sekundę. Šie rezultatai gauti realiuoju laiku gamybinėje mini- / mikro-LED linijoje. Tam, kad segmentacija būtų tiksli net esant nelygiam apšvietimui ar mechaninėms vibracijoms, įdiegta pagrindinių tikrinimo vienetų išskyrimo ir pasvirimo korekcijos sistema, leidžianti greitai atskirti defektines sritis.

Anomalijų duomenų trūkumo problema dažnai sprendžiama naudojant trūkstamų duomenų kūrimą. Defect-GAN (priešinius generatyvinius defektų tinklus angl. *defect-generative adversarial network*) modelis (žr. 3 pav.) generuoja realistiškus defektų pavyzdžius, leidžiančius papildyti treniravimo aibes ir padidinti defektų aptikimo modelių atpažinimo galimybes. Eksperimentuose parodyta [28], kad sukurti sintetiniai pavyzdžiai pagerina aptikimo tikslumą, nuo 92,7 proc. iki 96,9 proc., ypač kai realių defektinių pavyzdžių skaičius yra ribotas.



3 pav. Defektų generavimas „Defect-GAN“ tinklu, imituojant realias gamybines situacijas [28]

Naudojant tradicines vaizdų apdorojimo ir segmentacijos technikas, tokias kaip išankstinis pozicijos įvertinimas, modifikuotą normalizuotą koreliacijos koeficientą (angl. *normalized correlation*

coefficient, NCC) ir savybių sustiprinimo metodus, pasiekiamas 99,54 proc. tikslumas, 0,03 proc. aptikimo praleidimo rodiklis ir apie 1,1 ms vienos mikroschemos aptikimo trukmė, kurie užtikrina pramoninėje gamyboje reikalingą spartą ir tikslumą [29]. Kiti tyrimai [30] pateikia hibridinius algoritmus, kuriuose derinamas geometrinis skaičiavimas ir konvoliuciniai tinklai: pradžioje atliekamas greitas apytikslis defektų aptikimas, po to „įtartinų“ sričių detali analizė CNN pagrindu, pasiekiant 96,7 proc. AP ir mAP rodiklius, kurie yra 3,63 proc. didesni, nei taikant YOLOv5 algoritmą ir 10,39 proc. viršija YOLOv2 ir „mIoU“ metodų rezultatus.

Moksliniai tyrimai [31–33] atskleidžia, kad giluminio mokymosi metodai su duomenų plėtra, dėmesio mechanizmais ir įvairiais optimizuotais praradimo funkcijų deriniais suteikia galimybę aptikti itin mažus, silpnai išreikštus defektus net tada, kai duomenų aibė yra išbalansuota ar trūksta defektų pavyzdžių. Vienas iš praktinių pavyzdžių – mikro-LED inspekcija su YOLOv3-Dense algoritmu, leidusi pasiekti 33,69 proc. aukštesnį mAP nei įprasto CNN atveju, naudojant SMD LED, o įdiegus „DenseNet“ algoritmą pagerėjo mažų taikinių atpažinimas [33].

Taip pat reikšminga ir duomenų papildymo (angl. *data augmentation*) funkcija – „Defect-GAN“ sprendimas leidžia generuoti įvairių dirbtinių defektų duomenų, kurių trūksta realioje gamyboje, taip gerėja defektų aptikimo tinklų treniravimo efektyvumas ir universalumas [28].

Pramoninėje gamyboje aktualūs yra ir automatizuoti algoritmai (AOI, YOLO, „Defect-GAN“), leidžiantys realiu laiku taikyti optimizuotus sprendimus tiesiogiai gamybos linijoje, su minimaliu vėlavimu ir aukštu aptikimo tikslumu. Tai tampa įmanoma tiek diegiant optimizuotus CNN, tiek „Transformer“ pagrindu veikiančius modelius [34].

Apžvelgti LED, mini-LED ir mikro-LED testavimo sprendimai rodo, kad didžiausią tikslumą paprastai užtikrina ne vienas metodas, o kelių etapų analizės grandinė. Literatūroje pateikti metodai leidžia pasiekti 95,82 proc. AUROC anomalijų aptikime, 99,12 proc. mini- / mikro-LED defektų atpažinimo tikslumą, 90,4 proc. mAP50 taikant transformerių pagrindu veikiantį defektų aptikimą, 99,1 proc. defektuotų ir gerų vaizdų atskyrimą bei iki 125 kadrų per sekundę apdorojimo spartą realiojo laiko gamybinėse linijose. Šie rezultatai patvirtina, kad optinių elementų testavime svarbu ne tik pats klasifikatorius, bet ir vaizdo srities lokalizavimas, triukšmo mažinimas, geometrijos korekcija ir tinkamai parinktas požymių skaičiavimo būdas.

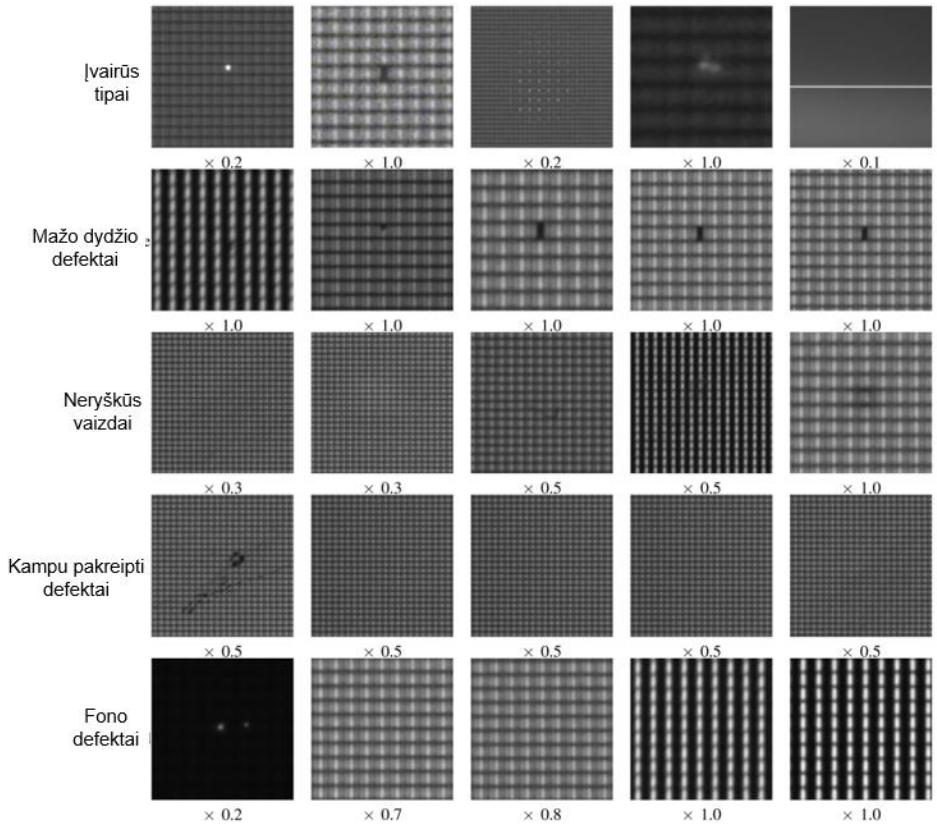
1.4. Skystųjų kristalų ekranų ir transporto LED modulių testavimo metodai

Šiuolaikinių transporto priemonių ir pramonės ekranų bei LED modulių testavimas susiduria su vis sudėtingesnėmis užduotimis, reikalaujančiomis ne tik tikslumo, bet ir spartaus apdorojimo bei patikimumo. Pastaraisiais metais vyrauja giliųjų neuroninių tinklų (angl. *deep learning*) metodai, išstumiantys tradicinius algoritmus dėl didesnio lankstumo, automatinio požymių išgavimo ir efektyvumo apdorojant įvairių tipų defektus ar signalus.

Transporto LED modulių, naudojamų, pavyzdžiui, optinės kameros komunikacijoje (OCC), testavimo problematika apima tikslų LED pozicijų nustatymą, būsenos atpažinimą ir imunitetą trikdžiams, tokiems kaip judesio suliejimas ar pašaliniai šviesos šaltiniai. Autoriai [35] pasiūlė sprendimą, pagrįsta modifikuotu YOLOv5 neuroniniu tinklu (D2Net), leidžiančiu vienu metu tiek aptikti LED masyvo pozicijas, tiek taikyti judesio suliejimo pašalinimą. Tai leidžia apdoroti duomenis realiu laiku be tikslumo praradimo, kadangi tinkle integruota defektų aptikimo ir paryškinimo funkcija. Be to, LED būsenoms atpažinti taikyta segmentacija ir konvoliucinis tinklas (CNN),

kuriame naudojamas FAST kampų detektorius ir masyvo blokų padalijimas į 256 regionus. Tokia strategija sumažina skaičiavimo apimtį ir leidžia vienu CNN klasifikatoriumi identifikuoti LED įjungimo / išjungimo būsenas net esant triukšmingoms aplinkos sąlygoms. Eksperimentuose patvirtinta, kad siūlomas metodas geriau susitvarko su judesio sukeliamu suliejimu nei ankstesnės alternatyvos: pavyzdžiui, SC (selektyvaus fiksavimo metodas angl. *selective capture*) metodo duomenų perdavimo sparta siekė 6,912 kbps, o klaidų dažnis 1,25 m atstumu tarp šviesos diodų masyvo ir kameros – 10^{-5}, tačiau turėjo ribotą pritaikomumą dėl didesnių skaičiavimų lyginant „Cam-Shift“ su *Kalmano* filtru ir prastesnio veikimo judant, o pasiūlytas D2Net metodas užtikrino aukštą tikslumą (98,6–99,2 proc.) ir veikė iki 10 m atstumu [35].

Testuojant skystųjų kristalų ekranų (angl. *liquid crystal display*, LCD) modulius susiduriama su kitomis problemomis – defektų spektras yra itin platus (žr. 4 pav.): nuo taškinių „šviesus taškas“, „tamsus taškas“ (angl. *bright spot*, *dark spot*) iki išplėstų plotinių ar neaiškių formų defektų, kurie gali būti aptinkami tik stebint juos tam tikru kampu ar naudojant skirtingus foninius vaizdus.



4 pav. LCD ekranų defektų tipai [36]

Mokslininkai [36] pasiūlė automatinės optinės inspekcijos (AOI) sistemą, kuri pritaikyta LCD modulių defektams pramoninėse linijose aptikti. Ši sistema sujungia kelių fonų (tamsus, baltas, pilkas, išjungtas) ir kampų (pagrindinis vertikalus ir keturi šoniniai) vaizdavimą, o vėliau taiko YOLOv3 detektorius kiekvienam defektų tipui ir atvaizdo konfigūracijai. Iš viso naudojami aštuoni YOLOv3 detektoriai, kiekvienas iš jų treniruotas atpažinti tam tikros klasės defektus konkrečioje aplinkoje. Prieš pateikiant vaizdus neuroniniams tinklams, jie yra apdorojami: pašalinami kraštai, didelės rezoliucijos nuotraukos suskaidomos į dalis (slankiojančio langelio metodu su persidengimu), tai leidžia išvengti defektų „praradimo“ dėl dalinio vaizdo atskyrimo. Be to, papildomai taikomi

specializuoti detektoriai, skirti triukšmui ir nereikalingoms detalėms, pvz.: dulkėms, plėvelėms, pašalinti. Tyrimų išvados rodo, kad pasiektas daugiau nei 95 proc. tikslumas, t. y. klaidingų atpažinimų yra mažiau nei 5 proc. [36].

Papildomai, transporto priemonėse plačiai taikomi mini-LED modulių testavimo metodai reikalauja didelio tikslumo ir gebėjimo atskirti defektus itin tankiose ir mažose struktūrose. Autoriai [37] naudojo aukštos raiškos neuroninį tinklą („Hrnet“), papildytą mišrių išretintų konvoliucijų (MDC-DUC) bei „likutinio globalaus konteksto dėmesio“ (RGCA) moduliais. Ši sistema geba detaliai segmentuoti atskirus mini-LED diodus ir tiksliai nustatyti defektinius taškus net esant jų dideliam tankiui ar silpnai išreikštoms riboms. Sistemos „mIoU“ pasiekė 86,91 proc. tikslumą, tai yra daugiau nei „Unet“, „PSPNet“, „Deeplabv3+“ ar bazinis „Hrnet“ tinklai, o vidutinis kvadratinis nuokrypis (RMSE) LED pozicijos ir defektų skaičiaus vertinime buvo mažesnis nei visų palyginamų tinklų. Modeliui treniruoti naudoti 2 118 skirtingų vaizdų su realiais ir modeliuotais defektais, o duomenims įsigyti naudota didelės raiškos CCD kamera (2448 × 2048 px, 3,45 µm × 3,45 µm). Toks derinys leido efektyviai aptikti ne tik mirusius (angl. *dead pixel*) taškus, bet ir neteisingai įstatytus ar išsidėsčiusius mini-LED modulius [37].

Galima pastebėti, kad transporto LED OCC sistemose daugiausia dėmesio skiriamas realaus laiko atpažinimo procesui įgyvendinti ir triukšmui judančiose scenose (naudojant D2Net, CNN ir papildomus segmentavimo metodus) valdyti, o LCD pramonėje pagrindiniai iššūkiai yra susiję su defektų duomenų bazės (vaizdų) sudarymu, vaizdų iš įvairių pozicijų bei itin didelės raiškos gavimu. Šiems probleminiams klausimams spręsti naudojamos kelių detektorių (YOLOv3) sistemos su išmaniųjų foninių vaizdų pasirinkimu bei duomenų į atitinkamas grupes suskaidymu. Mini-LED modulių atveju, didžiausią tikslumą lemia sudėtingi segmentacijos tinklai, gebantys išlaikyti didelės raiškos požymius bei taikyti globalios konteksto informacijos analizę, kas ypač svarbu, kai defektai yra subtilūs ar sunkiai pastebimi.

Apibendrinant, pažangūs optiniai testavimo metodai, paremti giluminiu mokymusi, užtikrina tiek transporto, tiek LCD ir mini-LED modulių kokybės kontrolės efektyvumą, o naujausios architektūros (YOLOv5, „Hrnet“, MDC-DUC, RGCA) leidžia pagerinti defektų aptikimo tikslumą, pritaikomumą realiose situacijose bei sumažinti klaidingų atpažinimų skaičių.

1.5. Spausdintinių plokščių (PCB) ir fotovoltinių panelių defektų aptikimo metodai

Fotovoltinių panelių defektų aptikimo metodų analizė pateikta [10] moksliniame šaltinyje, kuriame taip pat apžvelgtos pagrindinės vaizdų apdorojimo technologijos, taikomos fotovoltiniams moduliams. Autoriai nurodo, kad dažniausiai defektams aptikti naudojama infraraudonųjų spindulių (IR) termografija bei elektroliuminescencijos (EL) vaizdavimas. Nustatyta, kad, pavyzdžiui, taikant SLIC (angl. *simple linear iterative clustering*) superpikselių algoritmą, efektyviai segmentuojamos karštųjų taškų zonos termovizinėse nuotraukose (žr. 5 pav.), tai leidžia identifikuoti kontaktų ar medžiagų pažeidimus. EL vaizdai leidžia vizualizuoti mikroįtrūkimus ir delaminaciją, ko neįmanoma atlikti su RGB vaizdais. Nurodoma, kad skirtingoms defektų klasėms aptikti naudojami skirtingi vaizdo apdorojimo algoritmai: paviršiaus įtrūkimai geriau aptinkami EL, o karštosios zonos – IR vaizduose. SLIC metodo jautrumas siekia 96 proc., kai paprastas slenkstis neviršija 85 proc. Taip pat akcentuojama, kad RGB analizė netinkama subtiliems defektams, nes nėra pakankamai jautri pokyčiams, todėl visais atvejais rekomenduojama derinti kelias vaizdinimo technologijas [10].

5 pav. Fotovoltinių panelių termovizinė nuotrauka [10]

PCB defektams [11] aptikti naudojami šiuolaikiniai metodai, autorių lyginti su tradiciniais vaizdo apdorojimo, mašininio mokymosi bei giluminio mokymosi algoritmais. Moksliniame šaltinyje aiškiai aprašytos šiuo metu naudojamos CNN, YOLO, SSD architektūros, jų tikslumai ir ribotumai. Pvz., CNN pagrindu veikiantys sprendimai PCB defektams identifikuoti dažnai pasiekia 95–98 proc. tikslumą, priklausomai nuo defektų klasės ir duomenų kokybės. YOLO modeliai išsiskiria didesne greitaveika ir tinka realaus laiko inspekcijai – autoriai ištyrė, kad, pavyzdžiui, YOLOv3 modelis, apdorodamas gamyklos sąlygomis nufotografuotas PCB nuotraukas, atpažino pagrindines defektų klases 97 proc. tikslumu, o apdorojimo greitis siekė 50 kadrų per sekundę. Tačiau tyrėjai pažymėjo, kad visi giluminio mokymosi metodai reikalauja didelių, kokybiškai sužymėtų duomenų rinkinių, kitaip atsiranda klaidingų atvejų [11].

Straipsnyje [12] pateikiama išplėstinė PCB defektų aptikimo metodų apžvalga, pagrindinį dėmesį skiriant giliojo mokymosi modeliams ir jų palyginimui su tradicinėmis AOI sistemomis. Detaliai aprašomi tiek vaizdo apdorojimo (šablonų palyginimo, kraštų aptikimo, morfologinės operacijos), tiek CNN algoritmai. Nustatyta, kad CNN (pvz.: „ResNet“, „EfficientNet“) modeliai išlaiko aukštą 95–98 proc. tikslumą net esant duomenų iškraipymams ar apšvietimo pokyčiams, kai tuo tarpu tradiciniai metodai (šablonų atitikimas (angl. *template matching*), slenkstis) dažnai nesugeba atskirti mažų skylučių, įtrūkimų, trumpinimų tarp itin plonų takelių ar kompleksinių defektų. Eksperimentuose įrodyta, kad, pavyzdžiui, CNN modelis atviras grandines ir trumpinimus atpažįsta 98 proc. tikslumu, o šablonų palyginimo metodas – tik 81 proc. tikslumu [12]. Nurodoma, kad didžiausią pranašumą CNN duoda daugiasluoksniuose, sudėtinguose PCB, kur tradiciniai metodai nespėja su naujomis defektų variacijomis.

Dar vieni mokslininkai [15] tarpusavyje pagal tikslumą, apdorojimo laiką bei panaudojimų išteklių kiekį lygino populiariausius modelius (pvz.: YOLOv4, FPN, „MobileNet“, SSD). Eksperimentai parodė, kad YOLOv4 algoritmas pasiekęs 95 proc. (angl. *intersection over union*, IoU) tikslumą, išlaiko virš 97 proc. detekcijos tikslumą, o „MobileNet“ algoritmas – 94 proc. tikslumą, bet reikalauja mažesnio modelio dydžio (MB) ir tinka įterptinėms sistemoms. Pastebėta, kad modelių universalumas priklauso nuo treniravimo duomenų ir hiperparametrų optimizavimo, o skirtingos

architektūros tinkamos skirtingoms defektų klasėms. GAN tinklams reikalingas papildomas duomenų generavimas treniravimas [15].

Šaltinyje [38] analizuojamas visiškai konvoliucinių tinklų (angl. *fully convolutional networks*, FCN) taikymą LED mikroschemų defektams aptikti, naudojant fotoliuminescencijos (PL) vaizdus. FCN modeliai treniruoti su ribotu, bet kokybiškai sužymėtu duomenų kiekiu, sugebėjo klasifikuoti kiekvieną LED lustą pagal pažeidimą, taip pat aptikti tiek pavienius defektus, tiek defektų klasterius, kuriuos klasikiniai slenkstiniai (angl. *threshold*) ar taisyklėmis grįsti metodai praleidžia. Parodyta, kad naudojant FCN, modelio jautrumas siekia iki 98 proc., o klaidingų teigiamų atvejų skaičius sumažinamas iki 2 proc., kai taisyklėmis pagrįstų metodų klaidingi atvejai viršija 10 proc.. Akcentuojama, kad FCN ypač tinka gamybos linijų kokybės kontrolei, kai reikalingas visos matricos defektų žemėlapis.

Autoriai [16] apžvelgė automatizuotos PCB inspekcijos metodus ir jų evoliuciją per paskutinius 25 metus. Šaltinyje sistemingai aprašytos tradicinės AOI, „X-ray“, laminografijos, ultragarsinės, terminės analizės ir AI pagrindu veikiančios kompiuterinės regos metodikos. Pastebėta, kad didžiausiu pranašumu pasižymi AOI sistemos su giluminio mokymosi integracija, tačiau nurodyti ir pagrindiniai trūkumai: didelės diegimo sąnaudos, būtinas nuolatinis modelio adaptavimas, o silpnai žymėtų ar keičiamų duomenų atveju – sumažėjęs tikslumas. Taip pat nurodoma, kad pagrindinės gamybinės problemos kyla dėl „duomenų pasislinkimo“ (angl. *concept drift*), kai modelis dėl pasikeitusių sąlygų pradeda daryti klaidas. Straipsnyje pabrėžiama viešų PCB duomenų rinkinių svarba ir rekomenduojama nuolat testuoti modelių veikimą naujuose domenuose.

Tyrėjai taip pat [26] tyrė vaizdų analizės ir giluminio mokymosi algoritmų taikymą safyrinio epitaksinio pagrindo (angl. *sapphire epi-wafer*) defektams žalių LED gamyboje aptikti. Eksperimentuose analizuotas tradicinis vaizdų apdorojimas sujungtas su defektų detektavimo algoritmu. Rezultatai rodo, kad CNN pagrįsti modeliai, palyginti su taisyklėmis grįstais metodais, pasiekė 96 proc. defektų aptikimo tikslumą, kai tuo tarpu klasikiniai vaizdų analizės metodai – tik 84 proc. Taip pat pažymima, kad CNN modeliai leidžia automatizuotai atskirti ne tik aiškius, bet ir subtilius defektus, kas iš esmės padidina bendrą kontrolės patikimumą. Tyrime buvo taikyta G3-GM11-M2420 kamera (5 MP), optinis apšvietimas, o modelių treniruotė atlikta su 161 sužymėtu atvaizdu, apimančiu tris defektų tipus (dalelių, įspaudimų ir dislokacijos). CNN aptiko visų klasių defektus su 96 proc. tikslumu, kai taisyklėmis grįstų algoritmų tikslumas siekė 80–85 proc., ypač silpnėjant aptikimui, jei defektas netipiškas ar žemas vaizdo kontrastas.

Apibendrinant, mokslinių šaltinių analizė rodo, kad geriausi rezultatai pasiekiami taikant giluminio mokymosi metodus, ypač CNN ir FCN, kurie dažniausiai pranoksta tradicinius vaizdų apdorojimo metodus tiek tikslumu (skirtumas siekia 10–15 proc.), tiek gebėjimu identifikuoti sudėtingesnius defektus ir prisitaikyti prie naujų gamybinių situacijų. Efektyviems AI metodams pritaikyti būtinas kokybiškas duomenų žymėjimas, nuolatinis modelių adaptacijos užtikrinimas ir eksperimentinis validavimas realiose sąlygose.

1.6. Pažangūs ir netradiciniai optinių elementų kokybės vertinimo metodai

Pastaraisiais metais optinių elementų, ypač LED ir mikro-LED, kokybės vertinimo srityje ryškiai išsiskiria pažangūs automatiniai defektų nustatymo metodai, paremti giluminiu ir mašininiu mokymusi bei pažangiomis signalų ir vaizdų apdorojimo technikomis. Tradiciniai defektų aptikimo metodai, tokie kaip vizualinė inspekcija ar paprasti fizikiniai matavimai, pasižymi ribotu tikslumu,

didele subjektyvumo rizika ir žemu našumu esant masinei gamybai. Todėl šiame poskyryje didelis dėmesys skiriamas pažangiems algoritmams kurti, siekiant ne tik pagerinti aptikimo tikslumą, bet ir sumažinti laiko bei resursų sąnaudas.

Vienas iš svarbiausių žingsnių – visiškai konvoliucinių tinklų taikymas fotoliuminescenciniams vaizdams apdoroti, kai kiekvienas LED lustas automatiškai klasifikuojamas kaip defektuotas arba tinkamas. Šis metodas leidžia realizuoti pikselių lygio defektų klasifikaciją net ir tada, kai defektų pasiskirstymas duomenyse yra nesubalansuotas, nes svorių pritaikymas (angl. *weighted loss*) padeda sumažinti mokymo disbalanso poveikį. Atlikti eksperimentai parodė, kad tinklas geba aptikti tiek didelius defektų klasterius, tiek pavienius defektus, taip pat analizuoti nelygaus ryškumo vertes [38].

Papildomai, siekiant įveikti žymėtų duomenų trūkumą ir klasių disbalanso problemą, literatūroje [22] nagrinėti ir neprižiūrimieji defektų aptikimo būdai. Vienas tokių – auto enkoderių pagrindu veikiantis anomalijų detektorius, kuris reikalauja tik normalių pavyzdžių ir geba atskirti anomalijas pagal rekonstrukcijos klaidos dydį. Eksperimentai su mikro-LED duomenų rinkiniu parodė, kad šis metodas pasiekė 95,82 proc. AUROC (kiekybinis modelio gebėjimo atskirti klases įvertinimas, angl. *area under receiver operating characteristic curve*) rezultatą, t. y. 20,87 proc. geresnį nei įprastas konvoliucinis auto enkoderis ir 0,67 proc. – nei „Deep SVDD“, išvengiant duomenų žymėjimo būtinybės.

Kitas netradicinis būdas [20] – spektrinės galios pasiskirstymo (SPD) modeliavimas ir mašininis mokymasis, taikant tiek statistinius modelius („Gauss“, „Lorentz“, „Asym2sig“), tiek pagrindinių komponentų analizę (PCA) bei *K*-artimiausių kaimynų (KNN) metodą. Ši schema leidžia identifikuoti ankstyvas LED degradacijos stadijas, sumažinant anomalių šviesos diodų detekavimo laiką iki 789,6 val., palyginti su 1 311 val., kol šviesos srautas sumažėja iki 70 proc., kaip reikalauja tradiciniai IES TM-21 testai. Rezultatai parodė, kad „Asym2sig“ funkcija turi geriausią pritaikymo rezultatą LED spektrinėms charakteristikoms modeliuoti.

Pritaikant giluminius neuroninius tinklus, tokius kaip CM-YOLOv5 su „Multi-Head Self-Attention“ mechanizmu, 3,8 proc. buvo padidintas vidutinis tikslumas (angl. *mean average precision*) ir 3,7 proc. tikslumas (angl. *precision*), palyginti su pagrindiniu YOLOv5 algoritmu, taip pat viršyti kitų pagrindinių objektų detekcijos algoritmų rezultatai (pvz.: YOLOR, YOLOX, YOLOv6). Šio algoritmo pranašumas – efektyvus veikimas net esant dideliam duomenų nesubalansavimui bei diegimo galimybė įterptiniuose įrenginiuose, pvz., „NVIDIA Jetson Xavier NX“ plokštėje [34].

Išskirtinas ir hibridinis LED lustų defektų aptikimo algoritmas [30], jungiantis geometrinius skaičiavimus ir konvoliucinį neuroninį tinklą. Geometrinė dalis skirta pirminiam, greitam defektų filtravimui, o CNN – tikslesnei antrinei analizei. Šis metodas taikomas tūkstančiams LED lustų su mažiau nei 5 proc. defektų dažniu analizuoti, ir pasiekia 96,7 proc. vidutinį tikslumą (AP). Palyginti su tradiciniu YOLOv2 algoritmu gauta 10,39 proc. aukštesnė mAP reikšmė, o su YOLOv5 algoritmu – 3,63 proc. didesnė „mIoU“ reikšmė.

Pažymėtina ir „fuzzy“ patikimumo teorija [39], kuri gali būti taikoma LED signalinių šviestuvų patikimumui kelių koridoriuje įvertinti. Tradiciniai dvejetainiai patikimumo modeliai neefektyvūs sistemoms, kurios dirba su pereinamosiomis degradacijos būsenomis. „Fuzzy“ patikimumo metodai leidžia įvertinti tiek palaipsniui degradavusius, tiek iš dalies veikiančius LED blokus, remiantis „narystės“ (angl. *membership functions*) funkcijomis, apibrėžiančiomis laipsnišką perėjimą tarp „sėkmės“ ir „nesėkmės“ būsenų [39].

Taip pat išskiriamas duomenų praplėtimo (angl. *data augmentation*) metodas [28], pagrįstas „Defect-GAN“ – generatyvinių priešpriešinių tinklų taikymu aukštos kokybės ir įvairove pasižyminčių defektų pavyzdžių sintezei. Šis metodas leidžia imituoti defektus normaliuose vaizduose, kontroliuoti defektų vietą, kategoriją bei stochastinę įvairovę, kas ypač svarbu treniruojant inspekcijos tinklus, kai realių defektų pavyzdžių yra mažai. „Defect-GAN“ metodu sintezuoti duomenys leidžia pasiekti geresnį inspekcijos tinklų apmokimą ir aukštesnius kokybės rodiklius, palyginti su tradiciniu GAN tinklo ar riboto realių defektų duomenų panaudojimu.

Apibendrinant, pažangūs ir netradiciniai optinių elementų kokybės vertinimo metodai grindžiami giluminio mokymosi (FCN, CNN, YOLOv5, hibridiniai CNN), nesupervizuoto mokymosi (autoenkoderiai, KNN, PCA), statistinės analizės (SPD modeliai), „fuzzy“ patikimumo teorijos ir duomenų sintezės („Defect-GAN“) metodais. Šių sprendimų taikymas leidžia padidinti defektų aptikimo tikslumą ir greitaveiką, sumažinti žmogiškųjų klaidų riziką, efektyviai naudoti ribotus duomenų kiekius ir atlikti LED degradacijos bei defektų analizę ankstyvosiose stadijose ar sudėtingose realaus pasaulio situacijose.

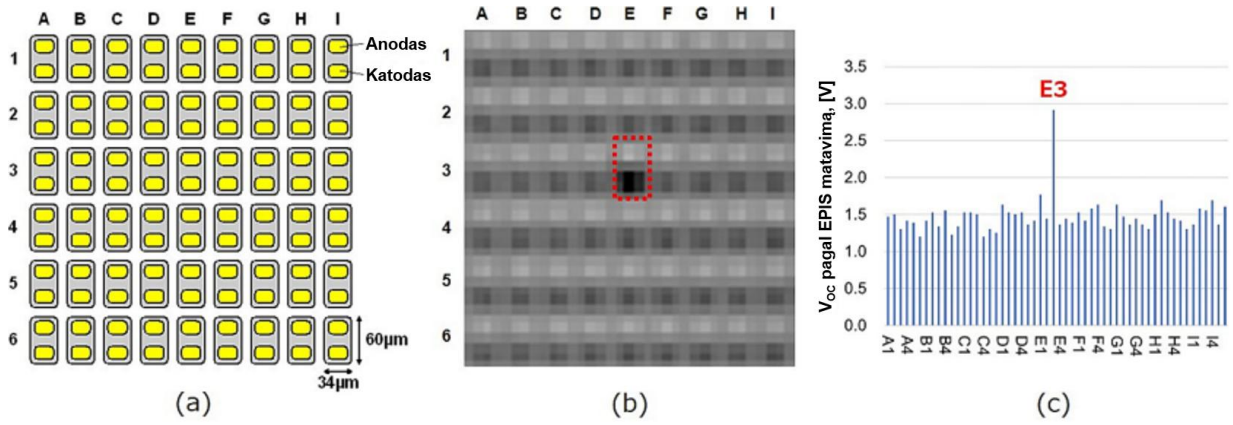
1.7. Fiziniai, eksploataciniai ir patikimumo aspektai optinių elementų testavime

Mikro-LED optinių elementų testavimas šiuo metu apima tiek fizinių savybių vertinimą, tiek eksploatacinių parametrų analizę bei patikimumo aspektus, kadangi šių elementų taikymas tiesiogiai priklauso nuo jų kokybės, patikimos eksploatacijos ir ilgaamžiškumo. Vienas iš esminių fizinių parametrų – mikro-LED lustų sujungimo stipris su pagrindu, kuris lemia viso ekrano struktūros stabilumą ir atsparumą eksploatacinėms apkrovoms. Iki šiol, dauguma tyrimų apsiribojo tik išorinių paviršiaus defektų identifikavimu, tačiau šaltinio [2] autoriai pasiūlė sistemą, leidžiančią kiekybiškai įvertinti mikro-LED lustų tarpusavio sujungimo jėgą. Pritaikyta optinio sverto jautrinimo technologija su mikro-jėgos jautrumo konsolėmis leidžia tiksliai išmatuoti lusto ir pagrindo sąsajos sujungimo jėgą. Sistemos kalibravimo ir matavimo eksperimentų rezultatai parodė, kad patikima vieno lusto sujungimo jėga turi būti ne mažesnė kaip 2 580 μN. Toks metodas leidžia identifikuoti tiek silpnus sujungimus, tiek galimus defektus, kurie tiesiogiai lemia gaminio patikimumą, nes per silpnas sujungimas sukelia šviesos netolygumą, spalvų poslinkius ar net elementų atsisluoksniavimą eksploatacijos metu. Taip pat nustatyta, kad vien pozicinių nuokrypių detekcija yra nepakankama, todėl būtina matuoti ir pačią sujungimo jėgą, o šis metodas užtikrina gamybos kokybės optimizavimą ir leidžia taikyti efektyvią defektinių elementų pakeitimo ar remonto strategiją.

Testavimo metu svarbus ir kontaktinis mikro-LED defektų nustatymas, tačiau gamybos efektyvumui užtikrinti ir siekiant, kad nebūtų pažeisti elementai, reikalingi bekontakčiai metodai. Šaltinyje [8] aprašytas didelės spartos, bekontaktis mikro-LED defektų tikrinimo metodas, pagrįstas fotoelektros efektu, išsiskiria tuo, kad leidžia vienu metu per maždaug 2 s įvertinti daugiau kaip 50 000 mikro-LED lustų (60 μm × 34 μm). Sukurtoje sistemoje naudojamas artimos talpos vaizdo jutiklis, kuris leidžia išmatuoti dėl fotoelektrinio efekto tarp anodų ir katodų susidarantį įtampų skirtumą (žr. 6 pav.), LED apšvietus trumpesnės bangos ilgio šviesa nei jų emisijos bangos ilgis. Svarbu, kad ši metodika nereikalauja fizinio kontakto su lusto elektrodais, todėl nesukelia žalos kontaktams, kas ypač svarbu miniatiūriniams lustams ir leidžia aptikti tiek elektros atjungimus, tiek trumpuosius jungimus. Toks metodas užtikrina ne tik aukštesnį testavimo greitį ir didesnį produktyvumą, bet ir didesnį tikslumą nustatant vidines jungtis, palyginti su klasikiniu kontaktiniu testavimu, kuris yra daug lėtesnis ir sukelia elektrodo pažeidimus. Eksperimentiškai įrodyta, kad EPIS (angl. *electrical*

picture inspection system) sistema užtikrina rezultatų atkartojamumą ir gali būti pritaikoma masinėje gamyboje.

Kitas svarbus eksploatacinis aspektas yra patikimumo LED sistemose vertinimas, ypač tuomet, kai LED naudojami kaip ilgaamžiai šviesos šaltiniai, pvz., transporto šviesoforuose. Šaltinyje [39] pateikiamas „fuzzy“ teorijos taikymas LED pagrindu veikiančių šviesoforų patikimumo analizei. Skirtingai nuo tradicinio vertinimo (veikia / neveikia), ši metodika leidžia analizuoti sistemų veikimą degradacijos būsenoje, kai dalis LED matricos jau pažeista, tačiau sistema vis dar funkcionuoja. Patikimumas matuojamas kaip laikas, per kurį šviesos srautas sumažėja iki 70 proc. pradinės vertės (L70), o realiame pavyzdyje („West Gate Boulevard, Austin“, TX) nustatyta, kad 11 proc. LED matricos buvo defektuotos jau tyrimo metu. Taikyta neryškiųjų aibių analizė leidžia tiksliai įvertinti tiek atskirų elementų, tiek visos sistemos patikimumą, kas svarbu transporto saugai ir aptarnavimo kaštams planuoti. Tyrimuose parodyta, kad LED eksploatacijos trukmė dažnai prognozuojama remiantis ribotais bandymų rezultatais ir tai gali lemti pernelyg optimistinius patikimumo įvertinimus, todėl rekomenduojama naudoti ilgesnio laikotarpio stebėseną ir realių eksploatacinių parametrų analizę.



6 pav. Mikro-LED lustų matavimų schema (a), EPIS vaizdas (b) ir atviros grandinės įtampos (VOC) rezultatai (c) [8]

Vertinant gamybinius aspektus, ypač mikro-LED technologijos diegimą, moksliniame šaltinyje [6] pasiūlyta lanksčios zondinės struktūros sistema, leidžianti aptikti ne tik pavienius defektus, bet ir užtikrinti, kad gamybos metu būtų identifikuojami visi defektiniai elementai, įskaitant netaisyklingus kontaktus, mikroįtrūkimus ir kitus pažeidimus, kurių nemato optinė inspekcija. Lankstieji zondai sugeba prisitaikyti prie įvairių paviršių, todėl užtikrinama tiksli ir pakartojama defektų detekcija net sudėtingos geometrijos paviršiuose ar esant minimaliems kontaktiniams plotams.

Apibendrinant, šiuolaikiniai mikro-LED testavimo metodai remiasi pažangiomis fizinėmis (pvz.: sujungimo jėgos matavimas, bekontaktis defektų nustatymas), eksploatacinėmis (pvz.: testavimo greitis, testavimo žala, masinės gamybos galimybės) ir patikimumo („fuzzy“ teorija, realaus laiko degradacijos analizė) analizės priemonėmis. Modernių sistemų taikymas leidžia ne tik išsamiai charakterizuoti kiekvieno lusto ar matricos būklę, bet ir efektyviai planuoti gamybos procesus, užtikrinant aukštą produktų kokybę ir ilgaamžiškumą.

1.8. Patentų ir inžinerinių sprendimų analizė optinių komponentų bandymuose

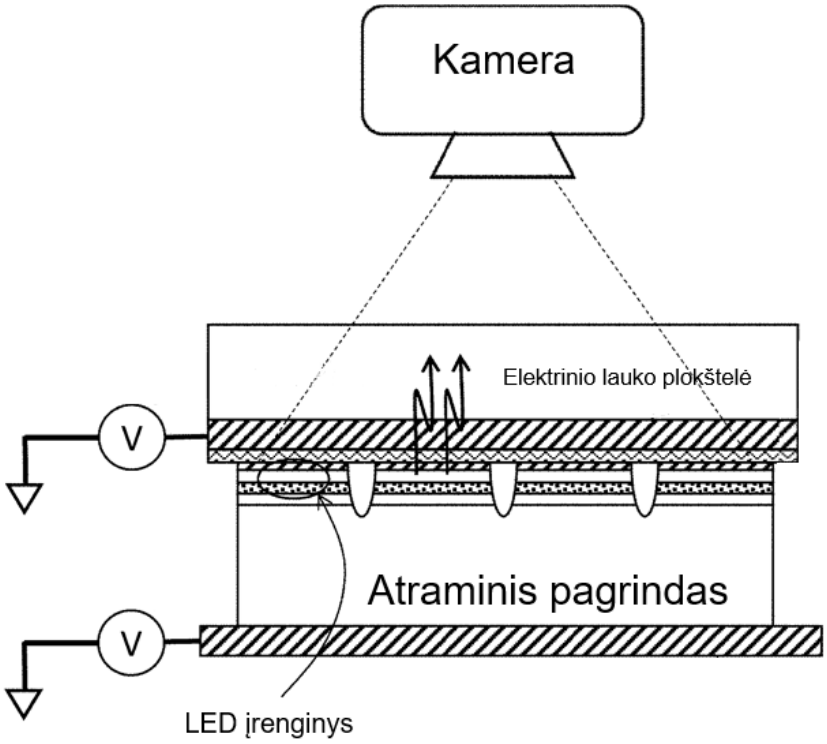
Analizuojant naujausius patentinius sprendimus vykdant optinių (ypač LED, mikro-LED) komponentų bandymus, matyti, kad pagrindinės inovacijos susijusios su testavimo metu fizinio kontakto eliminavimu, spektro filtravimu, viso proceso automatizacija, bei adaptacinių bandymų vykdymu.

Tyrėjas [3] pasiūlė metodą, kuriame LED matricos testuojamos naudojant talpinį srovės įvedimą be tiesioginio elektrinio kontakto (žr. 7 pav.). Naudojama dielektriku padengta elektrodo plokštelė (angl. *field plate*), kurios dielektriko storis tipiniame pavyzdyje yra nuo 1 iki 10 mikrometrų, dielektriko medžiaga – SiO_2, Al_2O_3 ar kita izoliacinė danga. Plokštelė priartinama 10–50 mikrometrų atstumu iki LED matricos paviršiaus. Į plokštelę per atskirą valdiklį (pvz., programuojamą funkcijų generatorių) paduodama įtampos nuo 0 iki 20 V rampa, kurios pakilimo laikas gali būti nuo 0,1 ms iki 200 ms, o tipinis dažnis – 10 Hz. Srovė į kiekvieną LED įvedama talpiniu būdu, bendras sužadintų LED skaičius priklauso nuo matricos: pavyzdyje minima 6 colių plokštelė, kurioje yra 250 000 LED elementų (po 250 µm kiekvienas) arba mikro-LED atveju – daugiau nei 40 mln. elementų (po 20 µm kiekvienas). Testavimo metu optinė emisija fiksuojama „CMOS“ kamera (rezoliucija tipinis pavyzdys – 5 MP), vaizdas apdorojamas programiniu būdu: kiekvienam LED nustatoma šviesos intensyvumo reikšmė. Rezultatai pateikiami kaip intensyvumo histograma (pvz., pavyzdyje parodytos 2100 atskirų kanalų), kuri pagal nustatytus slenksčius skirstoma į „PASS“ / „FAIL“. Norint įvertinti LED kokybę ar defektus, papildomai naudojami UV-IR bangos ilgio filtrai, pvz., „bandpass“ filtras 450–480 nm diapazonui. Privalumas – matavimai atliekami per visą plokštelę per vieną ciklą, nenaudojant mechaninių zondų. Kiekvieno ciklo trukmė priklauso nuo įtampos rampos – pvz., esant 200 ms rampai, visas 6 colių plokštelės testas trunka iki 10 s, priklausomai nuo vaizdo apdorojimo greičio. Jei lyginti su tradiciniu zondavimu (mechaniniais adatomis), kur kiekvienas LED užima 10 ms, visos matricos testas truktų virš 66 valandų.

Dar vieni autoriai [4] analizuoja specialų prietaisą LED plokštelėms (angl. *wafer*) testuoti, naudojant fotoliuminescencijos metodą. Testavimo metodika: per LED plokštelę paleidžiamas šviesos spindulys (pvz., 400–600 nm diapazono), pereinantis per LED sluoksnius. Už plokštelės įmontuotas spektrinis filtras, pavyzdžiui, 450–480 nm langui, užtikrina, kad detektorių pasiektų tik reikiamo diapazono šviesa. Detektorius – fotodiodas ar spektrometras, fiksuojantis šviesos srauto intensyvumą bei spektrą. Matavimo schema sudaryta iš: šviesos šaltinio (LED arba lazeris), optinės ašies (1–5 mm atstumu nuo LED paviršiaus), filtro ir detektoriaus. Testo metu automatiškai registruojami šviesos srauto pokyčiai, nustatomi emisijos stiprumas, bangos ilgis ir galimi nutekėjimo srovės pokyčiai, kurie ir parodo defektus. Metodas visiškai bekontaktis, skenuojant visą 4 arba 6 colių plokštelę matavimai trunka iki 1 min., priklausomai nuo sistemos konstrukcijos. Duomenys pateikiami kaip kiekvieno LED emisijos stiprumas, pagal kurį automatiškai identifikuojami brokuoti taškai.

Mokslininkas [5] pasiūlė modelį LED ekrano defektams aptikti. Jis savo sukurtoje sistemoje naudoja masyvinę LED matricos testavimo metodiką, užtikrinančią, kad duomenys apie šviesos emisiją ir elektrinius parametrus automatiškai įsikeltų į duomenų bazę. Modelis analizuoja kiekvieno LED šviesos intensyvumą, srovės ir įtampos charakteristikas, lygina jas su nustatytais normatyvais. Patvirtinta, kad LED, kurių IV charakteristika neatitinka normos (pvz., srovė prie 2 V mažesnė nei 10 µA arba šviesos intensyvumas mažesnis nei 50 proc. nuo nominalo), žymimi kaip defektuoti. Automatinė sistema veikia realiu laiku, duomenų apdorojimas užtrunka mažiau nei 1 s vienam 10

000 elementų moduliui, defektų nustatymo tikslumas priklauso nuo naudojamų kriterijų, tipinė klaidų norma – mažiau nei 0,5 proc., jei modelis apmokytas su tipine gamyklos statistika.



7 pav. Talpinė sąveika tarp elektrinio lauko plokštelės ir LED struktūros [3]

Šaltinyje [6] aprašyta lanksti mikro-LED testavimo adata, kurios skersmuo 5–20 μm, leidžiantis tiksliai pasiekti mikro-LED kontaktus, kai atstumai tarp elementų siekia 20–50 μm. Adata valdo pjezoaktuatorius, atliekančius vieno taško kontakto prispaudimą mažiau nei 0,1 N jėga, taip išvengiant kontakto pažeidimų. Testavimo metu registruojamos IV charakteristikos – kiekvienas mikro-LED įjungiamas nuosekliai, automatiškai fiksuojama srovė, įtampa, šviesos emisijos stiprumas. Defektai nustatomi pagal išskirtinius IV pokyčius: pvz., nutekėjimo srovė didesnė nei 1 μA (esant 2 V), arba jei šviesos intensyvumas mažesnis nei 30 proc. nuo vidutinio. Įranga leidžia testuoti iki 5 000 mikro-LED per valandą. Sistemoje naudojamas automatinis kontaktų nustatymo ir padėties koregavimo algoritmas, klaidos atveju adata automatiškai grįžta į pradinę padėtį ir testas kartojamas.

Tyrėjai [7] taip nagrinėjo LED ekrano serijinės jungties remonto metodą. Ekrane kiekvienas LED sujungtas nuosekliai, šalia įmontuoti atraminiai jungikliai (angl. *bypass switches*). Testavimo metu (prijungus testavimo įtampą nuo 0 iki 5 V, srovė iki 20 mA) nustatomas LED, kuris neveikia (t. y. nepraveda srovės arba nešviečia). Tuomet mikrovaldiklis automatiškai aktyvuoja šalia defektinio LED esantį jungiklį, kuris perjungia srovę per alternatyvią grandinę, taip užtikrinama, kad visas ekranas šviestų net ir su keliais defektais. Vieno defektinio elemento pašalinimas ir grandinės atkūrimas trunka iki 0,5 s, remonto tikslumas priklauso nuo LED matricos topologijos, tipinis išeigos (angl. *yield*) padidėjimas – 3–5 proc., kai brokuotų LED dalis neviršija 2 proc. nuo bendro kiekio.

Visuose šiuose sprendimuose pateikiami konkretūs parametrai, schemos, matavimo sekos ir automatinio testavimo algoritmai, skirti LED ir mikro-LED technologijoms masinėje gamyboje bei realiuose įrenginiuose testuoti.

1.9. Skyriaus apibendrinimas

Atlikta analizė parodė, kad elektronikos gaminių optinių elementų testavimo metodai vystėsi nuo lėtų, dažnai kontaktinių fizinių matavimų prie automatizuotų bekontakčių patikros sprendimų. Klasikiniai metodai, tokie kaip tiesioginis IVL charakteristikų matavimas ar kontaktinė zonduotė, tebėra svarbūs tada, kai reikia tiksliai įvertinti elektrinius, šiluminius ar spektrinius optinių elementų parametrus. Tačiau tokie metodai reikalauja stabilizuotų bandymo sąlygų, specializuotos įrangos ir ilgesnės matavimo trukmės. Literatūroje nurodoma, kad pilnas LED matavimo ciklas gali trukti 6–8 val., todėl masinės gamybos sąlygomis tokia patikra tampa našumo ribojančiu veiksniu [1].

Naujesni patentiniai ir inžineriniai sprendimai orientuoti į lygiagretų ir bekontaktį optinių elementų tikrinimą. Talpinės injekcijos, optinio filtravimo, lanksčių zondų ir bekontakčio fotovoltinio vertinimo metodai leidžia vienu metu tikrinti didelius LED masyvus, sumažinti tiesioginio kontakto poreikį ir sutrumpinti patikros trukmę nuo kelių valandų iki kelių minučių ar sekundžių [3–7]. Tokie metodai ypač aktualūs mikro-LED ir mini-LED gamyboje, kur tikrinamų elementų skaičius yra labai didelis, o mechaninis kontaktas gali sukelti papildomų defektų.

Automatizuotose kokybės kontrolės sistemose vis didesnę reikšmę įgyja vaizdų analizės ir giluminio mokymosi metodai. CNN, FCN, YOLO, „Transformer“ ir kitos architektūros naudojamos defektams aptikti, segmentuoti, klasifikuoti ir anomalijoms nustatyti. Apžvelgtuose tyrimuose tokie metodai dažnai pasiekia didesnį nei 95–99 proc. tikslumą, o kai kuriuose sprendimuose užtikrinamas ir realaus laiko veikimas, pavyzdžiui, 30–125 kadrų per sekundę apdorojimo sparta [11, 15, 18]. Palyginti su klasikiniais slenkstiniais, šablonų ar ribų aptikimo metodais, giluminio mokymosi sprendimai geriau prisitaiko prie sudėtingesnių vaizdų, nevienodo apšvietimo, skirtingų defektų formų ir gamybinių sąlygų variacijų.

Vis dėlto dauguma analizuotų sprendimų yra skirti ne paprastam optinio indikatoriaus būsenos nustatymui, o defektų paieškai, spektriniam įvertinimui, mikroskopinių struktūrų analizei arba didelio tankio LED masyvų kokybės kontrolei [20–24, 26]. Šiame projekte sprendžiamas siauresnis, bet praktikoje aktualus uždavinys – iš kameros vaizdo nustatyti segmentinio ekrano ir papildomų LED funkcines būsenas. Todėl išanalizuoti metodai šiame projekte nėra perimami tiesiogiai kaip galutiniai sprendimai, o naudojami pasirinktai tyrimo krypčiai pagrįsti: bekontaktis vaizdo fiksavimas, automatinis tiriamojo objekto lokalizavimas, kintančios geometrijos įvertinimas ir algoritminis optinių būsenų nustatymas.

Taigi 1 skyriuje atlikta analizė pagrindžia, kad pigesnė vaizdo kamera pagrįsta sistema gali būti tikslinga alternatyva tada, kai nereikia absoliučiai matuoti spektrinių ar elektrinių parametrų, o pakanka patikimai nustatyti optinių indikatorių funkcines būsenas. Toks sprendimas yra paprastesnis už specializuotas pramonines matavimo sistemas, tačiau jo patikimumas priklauso nuo vaizdo kokybės, apšvietimo, atspindžių, objekto padėties kadre ir algoritminio būsenų vertinimo stabilumo. Būtent šie veiksniai toliau nagrinėjami kuriant ir eksperimentiškai vertinant segmentinio ekrano bei LED būsenų atpažinimo algoritmą.

2. Vaizdų atpažinimo metodų taikymo elektronikos gaminių optiniams elementams testuoti galimybių analizė

Šio skyriaus paskirtis skiriasi nuo 1 skyriaus: pirmame skyriuje vertinama, kokie optinių elementų testavimo metodai egzistuoja ir kodėl specializuoti kontaktiniai ar spektriniai sprendimai nėra būtini siauram funkcinių „ON“ / „OFF“ būsenų uždaviniui. Šiame skyriuje nagrinėjami būtent vaizdų atpažinimo metodai ir jų tinkamumas pasirinktai sistemai. Palyginimas atliekamas pagal lokalizavimo galimybę, pakartojamumą, duomenų poreikį, jautrumą apšvietimui ir interpretavimo aiškumą.

Siekiant automatizuoti optinių elementų kokybės vertinimą ir sumažinti subjektyvius žmogaus-stebėtojo priimamus sprendimus, vis dažniau taikomos įvairios vaizdų atpažinimo technologijos. Jos leidžia ne tik fiksuoti defektus, bet ir analizuoti sudėtingus šviesumo, spalvos ir paviršiaus struktūros parametrus. Per pastaruosius ketverius metus giluminio mokymosi metodai pakeitė optinių elektronikos elementų testavimą iš esmės: požymių išgavimas ir atpažinimas sujungiami viename tinkle, todėl konvejeryje nebereikia atskirų aptikimo įrenginių. Literatūroje [9, 13–14, 17–18] fiksuotas vidutinis tikslumo pokytis nuo 78–82 proc. tradicinėse sistemose ir iki 91–99 proc. giliojo mokymo sprendimuose, o realaus laiko sistemose, kuriuose taikyti YOLO, DETR ar jų darinių algoritmai, apdorojimo sparta pakilo nuo ≈26 kadr./s iki ≥60 kadr./s. Duomenų sintezė naudojant DCGAN, „CycleGAN“ ar „Defect-GAN“ algoritmus leido sumažinti rankinio žymėjimo poreikį apie 50 proc. ir kartu iki 13 proc. padidinti absoliutų tikslumą. Šiame skyriuje nagrinėjama, kokiais konkrečiais parametrais (tikslumas, greitis, duomenų efektyvumas) vaizdų atpažinimo architektūros pralenkia klasikinę automatinę optinę inspekciją ir kokios sąlygos (vaizdo gavimo schema, modelio praretinimas, XAI diagnostika) būtinos, kad laboratorijose gauti rezultatai būtų atkartojami gamybinėje linijoje.

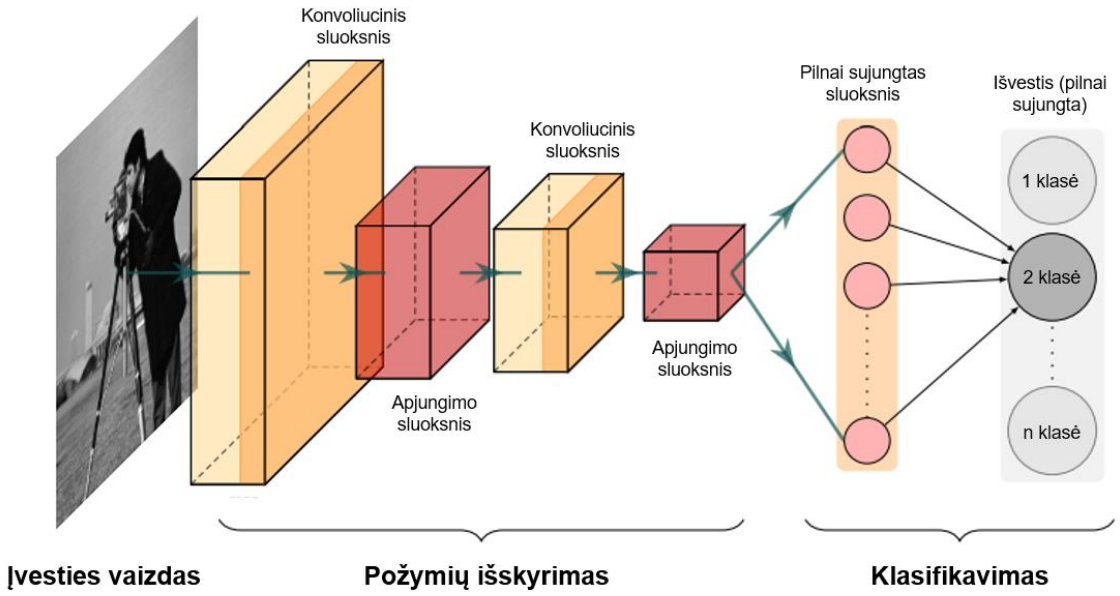
2.1. Dirbtinio intelekto technologijų įtaka optinių elementų testavimo pažangai

Dirbtinio intelekto (DI) taikymas elektronikos gaminių optinių elementų testavimo procese perėjo nuo tradicinių būdų, kuomet defektų filtrai parenkami rankiniu būdu, prie gilaus mokymosi modelių, kai vienoje grandinėje atliekama ir požymių atranka ir sprendimo apie defektus priėmimas. Pastebimas nuoseklus kokybės metrikų kilimas ir aiškiai išmatuojamas gamybos proceso optimizavimas.

Autorių apžvalga [9] rodo, kad modernūs CNN tinklai (žr. 8 pav.) ir jų dariniai jau pakeitė klasikinę automatinę optinę inspekciją: pvz., PCB paviršiaus defektams atpažinti taikomas „Faster R-CNN“ + FPN algoritmas pasiekė 95 proc. mAP reikšmę, o „Mask R-CNN“ ekranų matricų stiklo įbrėžimams – 94 proc. tikslumą, kai to paties objekto testavimo tradiciniai metodai neperžengdavo 80 proc. tikslumo. Autoenkoderiu grįstas LCD ekrano homogeniškų defektų radimas mažame duomenų rinkinyje parodė 100 proc. aptikimo rodiklį, bet autoriai pažymi, kad metodas kol kas tik eksperimentinis.

Dar vienoje apžvalgoje [14] pažymima, kad mišrus „CycleGAN“ + „U-Net“ algoritmas, sintezuojant defektus ir po to juos segmentuojant, įvairiose pramonės šakose stabiliai išlaiko ~95 proc. aptikimo ribą, panaikindamas pikselių žymėjimo būtinybę.

Mokslinėje disertacijoje [13] atlikti eksperimentai su WM-811K puslaidininkinių plokštelių žemėlapiais. Naudodamas DCGAN tinklą autorius kiekvieną iš aštuonių defekto klasių didino iki 10 000 vaizdų, 2 000 epochų mokymo cikle; 0,92 PCC (*Pearsono* koreliacijos koeficientas, angl. *pearson correlation coefficient*) riba leido atmesti ~50 proc. sintetinių kadrų ir išlaikyti duomenų švarą. „WaferCaps“ klasifikatorius su „CapsNet“ algoritmu (keturios vietoj dviejų konvoliucinių pakopų, 15 × 15 branduoliai vietoj 9 × 9), naudojant mišrųjį rinkinį parodė 99,59 proc. mokymo, 97,53 proc. validacijos ir 91,41 proc. testavimo tikslumą; dirbant tik su originaliais vaizdais testo rezultatai krito iki 78,2 proc., tad duomenų sintezė davė +13,21 proc. absoliutų tikslumo pagerėjimą.



8 pav. Konvoliucinių neuroninių tinklų (CNN) architektūra [13]

Vėliau tas pats autorius QCL bangolaidžių inspekcijoje sujungė „WaferCaps“ klasifikatorių su klasikiniu CNN tinklu: analizuotos trys klasės: „normali“, „užteršta“, „pažeista“ (angl. „normal“, „dirt“, „defect“), kiekvienoje po 1 000 vaizdų (800 tr./200 val.). Šis jungtinis modelis padidino bendrą sistemos tikslumą iki 98,5 proc., kai atskirai CNN tinklas ir „WaferCaps“ klasifikatorius pasiekia atitinkamai 97,2 proc. ir 93 proc. tikslumą. Šie skaičiai rodo, kad DI architektūrų hibridizavimas leidžia pasiekti 1,3–5,5 proc. didesnį tikslumą.

Autoriai [14] analizavo aparatūrinę ir programinę įrangą (angl. *hardware-software*): ciklinio konvejerio inspekcijoms jie rekomendavo naudoti linijinio skenavimo kameras, kurios kartu su „GigE Vision“ sąsaja leidžia realiu laiku perduoti viso pločio juostą, išvengiant buferio – to nepasiekia klasikinės ploto skenavimo (angl. *area-scan*) funkcijos, esant >10 m/min. gamybos greičiui. Taigi DI algoritmų našumas prasideda nuo tinkamai parinktos vaizdo fiksavimo sistemos.

Mokslininkų apžvalga [17], atlikta ekranų pramonėje akcentuoja, kad retų defektų nustatymo problema gali būti sprendžiama, atliekant privalomą geometrinių transformacijų ir spalvų erdvių keitimų augmentaciją, kol duomenų-kiekio-tikslumo kreivė išsilygina; jei balanso nepasiekiama, gali būti taikomi SMOTE ar kiti generatyviniai modeliai. Jie taip pat pateikė mokymosi eigos kreivę, kurioje terminalinis tikslumas sustoja didėti anksčiau kaip 50-osios epochos rate, todėl ankstyvas sistemos stabdymas išvengia perteklinio treniravimosi.

Tyrėjai [18] analizavo DI (dirbtinio intelekto) naudą gamybos linijų optinių elementų testavime: CNN pagrindu veikiančios AQA (automatinė kokybės užtikrinimo sistema, angl. *automated quality*

assurance), sistemos užtikrina linijinę analizę be papildomo mechaninio buferio, taip leidžiant koreguoti procesą „tą pačią sekundę“, nors konkretus vėlavimas straipsnyje nenurodomas. Autoriai pažymi, kad taikomi algoritmai eliminuoja žmogaus subjektyvumą ir leidžia palaikyti vienodus kokybės kriterijus keliose gamyklos vietose, kas ypač aktualu elektronikos gamyboje, kuri gali būti vykdoma per kelis cechus.

Apibendrinant, DI technologijos testavimo grandinėje naudojimo privalumai:

- tikslumas – iki 99 proc. laboratoriniuose ir 91–98 proc. realiuose testuose, kai tradiciniai metodai dažnai sustoja ties 80 proc. riba;
- duomenų efektyvumas – cikliniai GAN tinklai ir augmentacijos algoritmai leidžia sumažinti etikečių poreikį bent per pusę, o DCGAN tinklo naudojimas pagerina klasifikavimą daugiau nei 13 proc.;
- greitaveika – realaus laiko analizė be papildomo konvejerio stabdymo, ko nepasiekia rankinė inspekcija ar klasikinė vaizdų analizė.

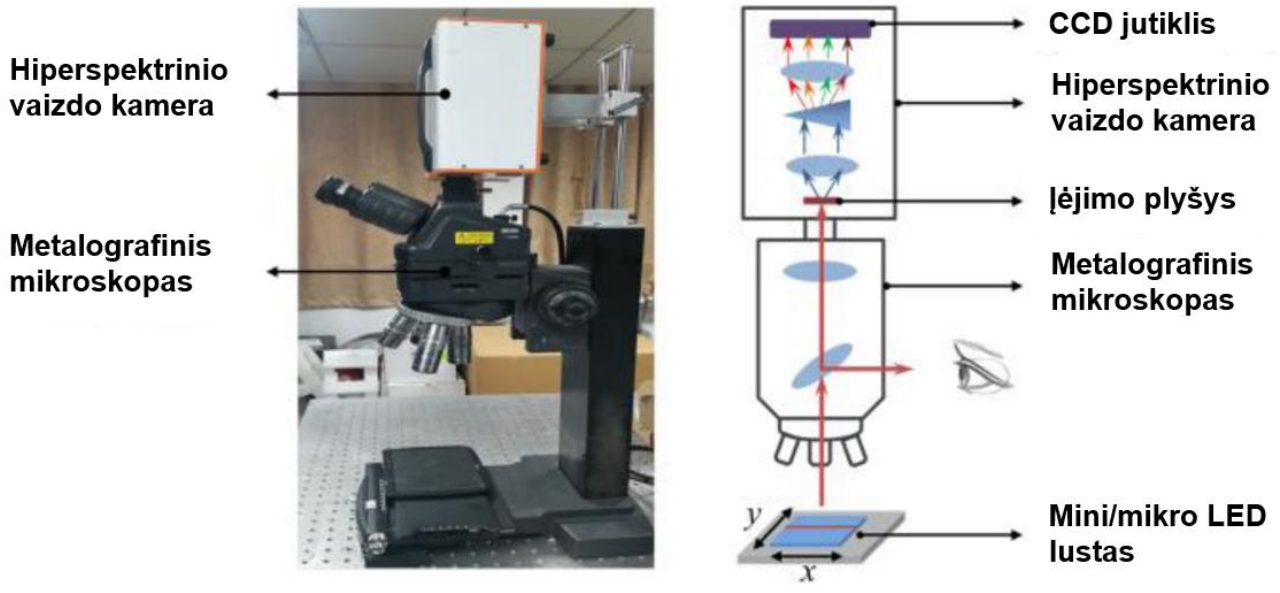
2.2. Modernios vaizdų analizės architektūros ir jų integracija į gamybos kokybės kontrolę

LED lustų tikrinimas vykdomas, taikant klasikinius konvoliucinius klasifikatorius. Pavyzdžiui, dvigubo erdvinio piramidinio srauto klasifikatorius PSPP-Net, apdorojęs 3 500 vaizdų, pasiekė 95,42 proc. tikslumą net esant ±20 proc. apšvietimo svyravimams, o pasukus lustus ±8° tikslumas sumažėjo tik 2,09 proc., t. y. 6,7 proc. geriau nei analogiškai treniruotas VGG-16 (gilusis konvoliucinis neuroninis tinklas, sukurtas Oksfordo universiteto angl. *visual geometry group*). Vienam LED ištestuoti reikia 12,4 ms, t. y. 1,6 kartų lėčiau, negu detektoriams, aptartiems [25] šaltinyje. Siekiant palaikyti linijos greitį, klasifikacija derinama su lokalia segmentacija. Trijų lygių lokalaus detalių stiprinimo karkasas (vaizdų apdorojimo metodas, kai visas vaizdas, vidutinio dydžio sritis, labai smulki lokali sritis sujungiami į vieną duomenų srautą, kuris perduodamas DI modeliui, (angl. *three-level local detail enhancement framework*), kuriame „LAF-Backbone“, CHMF ir „AH-Upsampling“ algoritmai pakeitė standartinį Mask R-CNN tinklą, mini- / mikro-LED rinkinyje mAP50 (vidutinis tikslumas, kai aptikimas laikomas teisingu, jei numatyta defekto vieta sutampa su tikra bent 50 proc.) reikšmė padidėjo nuo 92,8 proc. iki 97,38 proc., o mAP90 (vidutinis tikslumas, kai aptikimas laikomas teisingu, jei numatyta defekto vieta sutampa su tikra bent 90 proc.) – net 9,3 proc., t. y. iki 81,72 proc.; 2 048 × 2 048 kadras apdorojamas per 38 ms, leidžiant realiu laiku tikrinti 100 µm lustų juostą, kurios vaizdų srautas siekia 26 kadr./s [31].

YOLOv3-dense algoritmą papildžius „Dense-Block“ ir CIOU praradimo funkcijomis, mAP reikšmė padidėjo nuo 75,82 proc. iki 90,8 proc., o modelio parametrų kiekis sumažėjo 34 proc., todėl viso tinklo užimama atmintis tapo 1,5 kartų mažesnė nei originalo [33]. Tuo tarpu LBG-YOLO algoritmas, sujungęs LMCM, BiFPN metodus, tą pačią 90,8 proc. mAP ribą pasiekia per 125 kadr./s – beveik du kartus sparčiau nei „Faster R-CNN“ tinklas (65 kadr./s) [32]. MDCA-DETR, kuriame buvo pritaikyta deformuojama konvoliucija (filtrai prisitaiko prie objekto formos) ir koordinacinio dėmesio mechanizmas (leidžia tiksliai nustatyti objekto vietą vaizde), padidino AP50 nuo 86,2 proc. (pradinio DETR) iki 92,5 proc., o apdorojimo sparta siekė 62 kadrus per sekundę; ant GTX 3090 vienas kadras trunka 16 ms, taigi 62 kadr./s – 2,4 kartų sparčiau už pradinį DETR [24]. Naudojant pradinės padėties prognozę ir normalizuotą kryžminį koreliatorių galima per 1,098 ms, 99,54 proc. tikslumu patikrinti lusto kokybę, t. y. 8 kartus greičiau nei taikant klasikinį NCC tinklą, kurio tikslumas tesiekia 0,66 proc. [29].

Tais atvejais, kai defektų pavyzdžių beveik nėra, pranašumą įgauna neprižiūrimasis mokymas: mini-LED lustų blokų (256 × 256) testavimo tikslumas siekia 91,2 proc. AUROC ir užtrunka 18 ms–3,7 kartų greičiau nei to paties dydžio „U-Net“ autoenkoderis [22]. „STAE-Net“ (autokoderio pagrindu veikiantis neuroninis tinklas angl. *self-teaching autoencoder network*), apmokomas tik su nebrokuotais vienetais, rodo 99,1 proc. vidutinę „Image-ROC“ parametro reikšmę ir 125 kadr./s greitaveiką, lenkdama „FastFlow“ algoritmą atitinkamai 2,8 proc. ir 3 kartus [27]. Dar vienas būdas sumažinti defektų žymėjimų skaičių – sintetinių defektų generavimas, naudojant „Defect-GAN“ tinklą ir „ResNet-34“ klasifikatorių, šiuo atveju sistemos tikslumas pasikeitė nuo 70,25 proc. iki 75,48 proc., esant 65,6 FID rodikliui [28]. Siekiant užtikrinti ne tik tikslumą, bet ir stabilią klasifikavimo dispersiją, gali būti taikomas „Dual-Entropy“ CNN metodas su dviem skirtingomis divergencijos šakomis, kuris užtikrina 99,12 proc. tikslumą bei 3,5 ms/vnt. laiką, t. y. 3,1 proc. taškų daugiau ir 1,4 kartų sparčiau naudojant vien tik „ResNet-50“ metodą [23]. Savidėmesio mechanizmu pagrįstas CNN padidino maksimalų F1 rodiklį nuo 89,2 proc. (naudojant YOLOv5s architektūrą) iki 94,9 proc., nors vieno vaizdo apdorojimo trukmė išaugo 31 proc. – nuo 21 ms iki 27 ms [34].

Vaizdinę informaciją vis dažniau papildo spektrinė: mikroskopiniu hiperspektriniu matuokliu (žr. 9 pav.) sujungtas kraštų lokalizavimo metodas (optimizuotas *Canny*) 200 µm × 100 µm pikseliams pakėlė „mIoU“ nuo 88,49 proc. iki 97,36 proc., o fotometrinių parametrų atskirtis nuo integruojančios sferos liko 3,19 proc. ribose; nors vienam pikseliui reikia 0,14 s, vienu nuskaitymu apdorojamas 696 × 700 × λ duomenų kubas, todėl metodas tinka serijinei, ne gamybinės linijos analizei [21]. Tokios spektrinės sistemos gamykloje dažnai dirba kaip galutinė išeinančios kontrolės stotis, kai linija jau atmeta didžiąją broko dalį.



9 pav. Hiperspektrinio vaizdavimo sistema [21]

Šiuolaikinės patikros gamybos linijoje sistemos dažniausiai diegiamos GPU-FPGA hibridinėse matricose: optimizuotos YOLO ar MDCA-DETR architektūros, išlaikydamos ≥60 kadr./s, sinchronizuojamos su 50 000 vnt./h pernešančiais robotais, o neprižiūrimieji greitieji rekonstrukcijos tinklai paleidžiami FPGA antriniame sraute, kad rastų retas anomalijas. Praktikoje tai leidžia praleistų defektų rodiklį sumažinti žemiau 0,1 proc., tuomet vienam LED tikrinti sugaištama iki 2 ms, o

anotavimo poreikis sumažėja iki nulio, kai taikoma anomalinio mokymo strategija ar generuojami sintetiniai pavyzdžiai. Taip modernios vaizdų analizės architektūros, palyginti su 2019 m. standartu YOLOv3, užtikrina ne mažiau kaip 15 proc. geresnį mAP rodiklį ir bent du kartus spartesnį apdorojimą, neprarandant kokybės kontrolės tikslumo.

2.3. Automatinio defektų aptikimo sprendimai skirtingiems optiniams komponentams

Automatizuoto defektų aptikimo tyrimai skirtinguose optiniuose komponentuose per pastaruosius ketverius metus parodė, kad giluminio mokymosi architektūros gali pakeisti tradicinius šabloninio palyginimo metodus visoje gamybos grandinėje. Mini- / mikro-LED lustų gamyboje mokslininkai [31] pasiūlė segmentavimo tinklą, papildytą lokalaus detalumo stiprinimu; pramoninėje 4k × 6k vaizdų aibėje jie pasiekė 97,38 proc. mAP50 reikšmę ir sutrumpino vieno vaizdo analizės laiką 1,8 karto, palyginti su „RetinaNet“ algoritmu. Chen‘as [40] sukūrė AMS-YOLO algoritmą, kuriame savi dėmesio ir daugiasluoksnio suliejimo moduliai padidino mAP rodiklį 3,0 proc., tikslumą 3,5 proc., jautrumą 6,0 proc., o apdorojimo spartą 4,3 proc. palyginti su YOLOv5s algoritmu. Dar viename straipsnyje Han‘as ir komanda [32] sujungė LMCM + BiFPN + GDCA metodus. Ši kombinacija leido užtikrinti 90,8 proc. mAP rodiklį VOC2017 rinkinyje, taip sumažinus 32 proc. modelio parametrų, palyginti su „EfficientDet-D0“ algoritmu. Zheng‘as ir kt. [30] pasiūlė geometrijos filtravimu praturtintą SPP-YOLO metodą: 73 000 lustų plokštėje defektai aptikti 96,7 proc. tikslumu, tai 10,39 proc. geriau nei taikant bazinį YOLOv2 algoritmą, o vieno lusto analizės laikas sumažėjo nuo 15 ms iki 9 ms, t. y. 40 proc.

Optinės „mini / mikro-LED“ partijų charakteristikos matuojamos analizuojant ne tik vaizdą, bet ir spektrinės galios pasiskirstymą. Autoriai [21] sujungė mikroskopinę hiperspektrinę vaizdavimo sistemą su optimizuotu *Canny* algoritmu: 200 μm × 100 μm lustams išmatuotos EQE (išorinis kvantinis naudingumo koeficientas, angl. *external quantum efficiency*) ir (x, y) koordinatės skyrėsi ne daugiau kaip 3,2 proc. nuo integruojamojo sferos metodo, o 384 × 256 pikselių kadras analizuotas per 0,44 s.

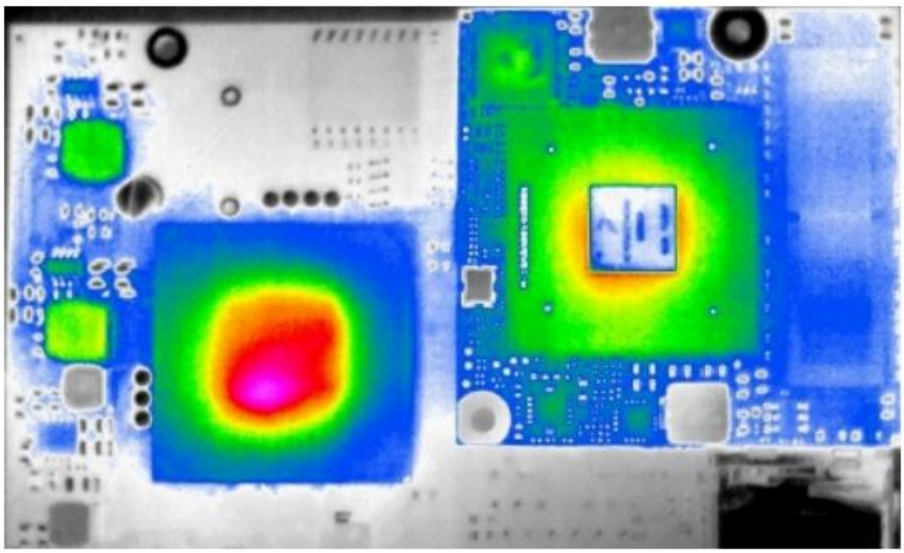
Autoriai [38] tyrė visiškai konvoliucinį „U-Net“ tinklą, apmokytą tik iš fotoliuminescencijos žymų; tinklas pasiekė 96,1 proc. tikslumą ir sumažino analizės laiką dvylika kartų, palyginti su tradiciniu elektriniu zondavimu. Tuo tarpu tyrėjai [26] OLED (organinis šviesos diodas, angl. *organic light emitting diode*) safyrinio epitaksinio pagrindo (angl. *sapphire epi-wafer*) defektus išrūšiavo taikant CNN tinklą, jie pasiekė 92 proc. teisingų aptikimų ir 1 proc. klaidingų signalų, kai tuo tarpu klasikinė morfologinė schema pateikė atitinkamai 71 proc. ir 7 proc.

Ankstyvosios degradacijos diagnostikai mokslininkai [20] taikė spektrinės galios pasiskirstymo (SPD) modelį: „Gaussian“, „Lorentz“ ir „Asym2Sig“ parametrų rinkinys sumažintas PCA iki dviejų komponentų, o vėlesnis KNN klasterizavimas sutrumpino privalomą testavimo trukmę nuo 1 311 val. (liumenų nuosmukis iki 70 proc.) iki 789,6 val. – tai 39,8 proc. laiko ekonomija be klaidingų signalų.

Didelėms panelėms ir optiniam kamerų ryšiui (OCC) analizuoti autoriai [35] pritaikė modifikuotą YOLOv5 algoritmą su judesio suliejimo kompensavimu: 640 × 480, 30 FPS sraute LED koordinatės apskaičiuotos per 14 ms, o klaidingai dekoduotų bitų dalis sumažinta nuo 7,9 proc. iki 1,6 proc. Heldt‘as [19] sukūrė unikalų papildymą „OpenCV“ programoje 64 × 32 LED panelėms; automatiniu ekspozicijos režimu teisingai įvertinta 99,4 proc. šviesos diodų. Į LCD modulių apžiūrą mokslininkai [36] įtraukė YOLOv3 algoritmą ir keturių pakopų filtravimą, leidusį identifikuoti 27 defektų klases – pasiektas 96,3 proc. F1 ir 680 mm/s konvejerio greitis, t. y. 6,5 karto greičiau už rankinę patikrą.

Tyrėjai [37] automobilių mini-LED foniniam apšvietimui pritaikė „HRNet“ tinklą su išretintais sluoksniais, koordinatės nustatytos su 0,14 mm RMSE paklaida – tai 18 proc. geresnis rezultatas, palyginti su „Deeplabv3+“ metodu.

Singh‘as ir kt. [11] nustatė, kad „Golden-board“ metodas atpažįsta iki devynių PCB defektų tipų (žr. 10 pav.), bet reikalauja 0,01 mm suderinimo; giliojo mokymo sistemos viešuose rinkiniuose viršija 95 proc. tikslumą, kai IoU reikšmė lygi 0,5. Autoriai [12] parodė, kad į „Transformer“ (dirbtinio intelekto modelio architektūra) pagrįstus detektorius įtraukus GAN sintetiką mAP reikšmė didėja iki 98,4 proc. „tuščios plokštės“ (angl. *bare-board*) scenarijuose. Chen‘as ir kt. [15] savo analizėje apžvelgė 30 tyrimų ir nustatė, kad realaus laiko linijoje „EfficientDet-D3“ algoritmas (640 × 640) analizuoja kadrą per <40ms. Kiti tyrėjai [16] išskyrė minėto koncepto trūkumą: jei modelis nepertreniruojamas, jo mAP reikšmė per metus sumažėja apie 6 proc.; autoriai pasiūlė papildyti analizuotą metodą aktyvaus mokymo schema su periodiniu perkrovimu.



10 pav. PCB plokštės termovizinė nuotrauka [11]

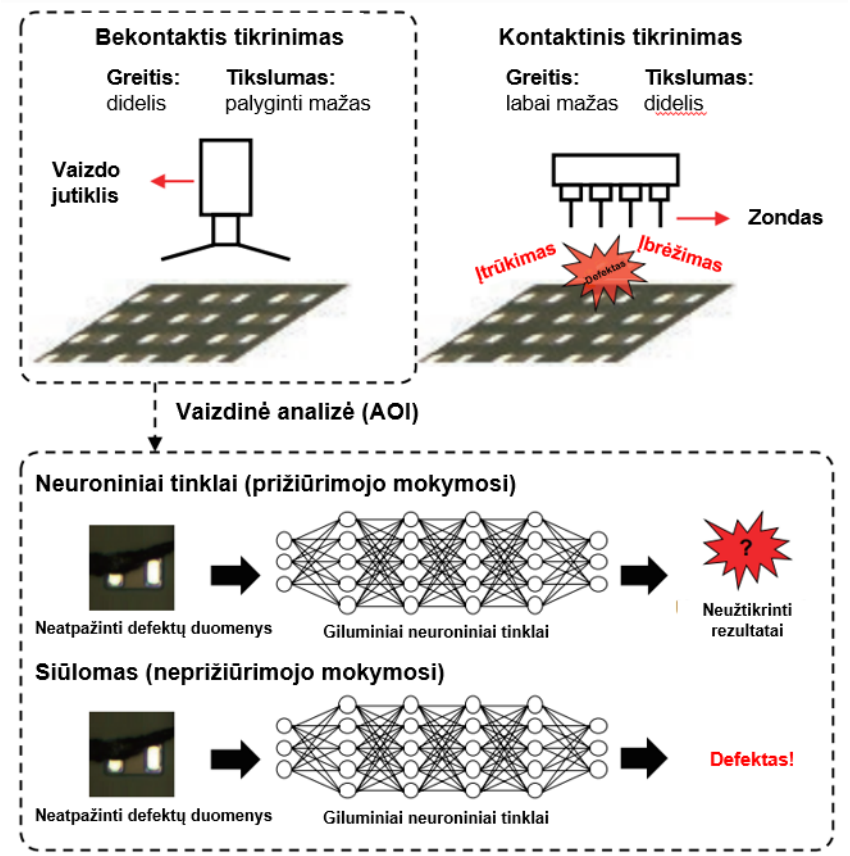
Afifah‘as ir kt. [10] apžvelgė 57 straipsnius apie saulės fotovoltinius modulius ir nurodė, kad vien EL (elektroliuminescencinės analizės) kanalai užfiksuoja 82 proc. paviršiaus įtrūkimų, o kombinuotas EL-IR (elektroliuminescencinės ir infraraudonosios analizės) metodas – 94 proc., taigi aptikimas pagerėja 12 proc.

Aukščiau pateikta mokslinių šaltinių analizė rodo, kad giliojo mokymo architektūros ne tik nuosekliai nuo 3 proc. iki 10 proc. didina mAP reikšmę palyginti su ankstesniais CNN tinklais ar kitais klasikiniais metodais, bet ir sutrumpina tikrinimo laiką nuo valandų iki milisekundžių vienam komponentui. Spektriniai ir fotoliuminescencijos modeliai leidžia anksti identifikuoti lusto degradaciją. Taigi, siekiant užtikrinti gaminamos produkcijos kokybę, rekomenduojama optinėms komponentams analizuoti taikyti savidėmesio arba „transformer“ tipo tinklus, kurie palaiko aktyvaus mokymo ir duomenų sintetinimo funkcijas.

2.4. Naujos kartos hibridiniai ir generatyvūs modeliai vaizdams atpažinti

Sparčiai mažėjant LED ir kitų optinių komponentų struktūroms, klasikiniai konvoliuciniai tinklai skaitmeninei vizualinei patikrai nebetinka, būtent dėl to pradedami naudoti naujos kartos hibridiniai bei generatyvūs modeliai.

Vienas iš tokių modelių – tai likutinis konvoliucinis automatinis kodavimo įrenginys (angl. *residual convolutional auto-encoder*, RCAE), t. y. anomalijų detektorius (žr. 11 pav.), kuris gali būti pritaikytas 4 629 mikro-LED vaizdų rinkiniui, kuriame defektų dalis tesiekia 2 proc. Geriausias modelio AUROC pasiekė 95,82 proc., t. y. 20,87 proc. daugiau už klasikinį CAE metodą ir 0,67 proc. daugiau už Deep SVDD algoritmą [22]. Nežymėto mokymo schema realiai eliminuoja klasių disbalanso problemą, o likutinės jungtys sumažina rekonstrukcijos paklaidas ir pagerina latentinės erdvės atskyrimą.



11 pav. Mikro-LED lustų gedimų aptikimas optiniu ir kontaktiniu būdais [22]

Kitas galimas sprendimas – semantinis segmentavimas fotoliuminescencijos (PL) vaizduose. Visiškai konvoliucinis tinklas, apmokytas su 145 plokštelėmis (iš viso ≈ 1,3 × 10^5 lustų), pasiekė 97,9 proc. pikselių tikslumą, 95,0 proc. vidutinį pikselių tikslumą ir 78,5 proc. „mIoU“ reikšmę; defektinių klasių atpažinimas siekė 88,7 proc. [23]. Gautus duomenis palyginus su rezultatais, gautais taikant „ResNet-101“ segmentavimo tinklą, pikselių aptikimo tikslumas buvo 3,1 proc. aukštesnis, o tinklo parametrų skaičius ~2 kartus mažesnis, todėl metodas tinka „inline-PL“ linijoms kai nėra naudojami galingi GPU procesoriai.

Hibridinėse architektūrose gali būti taikomas DENC-CNN tinklas, kuris 32 × 32 tašk. pilkų atspalvių vaizduose, „mini / mikro-LED“ lustų analizėje mažina mažos raiškos ir menkų defektinių klasių skirtumų sukeliamą informacijos trūkumą. Tiriant 451 nuotraukas pasiektas 99,11 proc. bendras sistemos tikslumas, 97,86 proc. *G*-mean ir 97,87 proc. F1 reikšmės, o klaidų rodiklis (RMSE) siekė tik 0,35 proc. [23]. Šis algoritmas 2,5 proc. tikslesnis nei klasikinis VGG-16 tinklas, nors parametrų skaičius jame 38 proc. mažesnis.

Dar viena dviejų pakopų schema siūlo sujungti geometriją (defekto formos, kraštinių ar kitų erdvinių parametrų išskyrimas) ir CNN tinklą. Pirmajame etape tinklelių segmentavimo būdu pašalinama ≈90 proc. nereikšmingų regionų, o antrajame pagerintas SPP-Net algoritmas fiksuoja likusias mikro defektų zonas. Vidutinė AP reikšmė siekė 96,7 proc.; tai 10,39 proc. geriau nei taikant standartinį YOLOv2 algoritmą, „mIoU“ reikšmė – 3,63 proc. geresnė nei analizuojant YOLOv5 algoritmą. Vieno vaizdo diagnostika užtruko 43 ms [30].

Į detektorių galvutes integruota savidėmesio funkcija (CM-YOLOv5) leidžia papildomai pagerinti tikslumą, kai mikro-LED matricos tankiai išdėstytos. Transformerių MA-CSP modulio naudojimas leido pakelti bazinio YOLOv5s algoritmo mAP reikšmę nuo 86,9 proc. iki 90,7 proc. (+3,8 proc.), preciziškumą – nuo 89,7 proc. iki 93,4 proc., o kadrų dažnį – nuo 78 iki 91 FPS [34]. Tai reiškia, kad identiškame įterptinės „Jetson Xavier NX“ procesoriuje sistema išlaikė >25 FPS greitaveiką realiuoju laiku, tuo tarpu klasikinė YOLOv5 algoritmo konfigūracija tesugebėjo užtikrinti 15–18 FPS greitaveiką.

Dar vienas moksliniuose straipsniuose nagrinėtas algoritmas – „Defect-GAN“. Jis sintezuoja aukštos kokybės defektinius vaizdus: FID (*Frécheto Inceptiono* atstumas, angl. *fréchet inception distance*) sumažėjo iki 13,7 (bazinis 31,2), o „ResNet-34“ klasifikatoriaus tikslumas su 50 000 sintetinių vaizdų pakilo nuo 70,25 proc. iki 75,48 proc. (+5,23 proc.) [28]. Sintetinių pavyzdžių injekcija ypač naudinga retoms klasėms, kurios realiuose rinkiniuose sudaro <1 proc. visų įrašų.

Šalia vaizdinių metodų, LED patikroje naudojami spektriniai modeliai bei patikimumo analizės sprendimai. Kaip pavyzdys, SPD + SVM algoritmas, apdorojantis iki 256 bangos ilgio vektorių. Jis nesužymėtas LED anomalijas aptinka iki 1 000 val. greičiau nei šviesos srauto testas, klasifikacijos tikslumas šiuo atveju siekia 97,5 proc. [20]. Neaiškiojo patikimumo (angl. *fuzzy reliability*) teorijos taikymas transporto šviesoforų LED lempoms analizuoti rodo, kad vos 15 proc. ryškumo nuokrypis sumažina patikimumo indeksą nuo 0,92 iki 0,71, o hibridinis trapecinis narystės funkcijų (angl. *hybrid trapezoidal membership function model*) modelis leidžia prognozuoti >90 proc. lempos tarnavimo laiką be fizinio ardymo [39].

Apibendrinant, hibridiniai (geometrija + CNN, entropija + CNN, savidėmesys + YOLO) metodai demonstruoja mAP reikšmės augimą nuo 3,8 proc. iki 10,39 proc., palyginti su vienpakopėmis klasikinėmis architektūromis, o generatyviniai modeliai – iki 5 proc. didesnį tikslumą. Unsupervised RCAE (neprižiūrimas likutinis konvoliucinis autoenkoderis, angl. *Unsupervised Residual Convolutional Auto-Encoder*) metodas užtikrina beveik 96 proc. AUROC be jokių anotacijų, todėl tinka ankstyvame gamybinių linijų diegime. Pastebėta, kad konvergencija, t. y. savi dėmesio blokai, entropijos reguliavimas ir GAN sintetiniai pavyzdžiai vis dažniau jungiami į vieną algoritmą, pasiekiant, kad mAP reikšmė būtų >90 proc., o FPS >60, t. y. atitinkančią realaus laiko elektronikos gaminių optinių elementų bandymų reikalavimus.

2.5. Praktiniai automatizuotų kokybės vertinimo sistemų diegimo aspektai

Skyriuje nagrinėjant praktinius automatizuotų kokybės vertinimo sistemų diegimo aspektus optinių elektronikos gaminių elementams, pirmiausia būtina atkreipti dėmesį į du svarbius dalykus – duomenų bazės paruošimą ir realaus laiko architektūros naudojimą. Mokslininkas savo disertacijoje [13] įrodė, kad „WaferCaps“ klasifikatorių rankiniu būdu treniruojant WM-811K aštuonių defektų klasių duomenų baze (kiekvienoje klasėje iki 10 000 vaizdų, apdorotų DCGAN tinklu), sistemos tikslumas padidėja iki 91,4 proc., t. y. 10,8 proc. daugiau negu taikant bazinį „CapsNet“ tinklą, kurio tikslumas siekia 80,6 proc. Tarp atskirų klasių didžiausias rezultatų pasikeitimas pastebėtas įbrėžimų kategorijoje – F1 rodiklis padidėjo nuo 0,66 iki 0,86, kai kartu taikytas rotacinis defekto klijavimas.

Tuo tarpu Islam‘as ir bendraautoriai [14] analizavo specialiai sukurtus modelius, kuriuose naudotas papildomas duomenų praretinimas bei kvantavimas. Gautas rezultatas – vėlavimo trukmė sumažinta iki 5 ms. Pritaikius praretintą „DenseNet“ algoritmą nedidelės galios ARM modulyje pavyko pasiekti 25 vaizdų per sekundę apdorojimo spartą, parametrai sumažinti 68 proc., tačiau 1,2 proc. sumažėjo tikslumas, palyginti su pilna versija.

Kameros ir apšvietimas dažnai nulemia bendrą sistemos patikimumą. Heldt‘o tyrime [19] parodyta, kad 64 × 32 LED plokštės patikrai pakanka paprastos „Logitech“ C920 internetinės kameros. Nustačius 0 proc. ekspoziciją ir 245–255 dvipusį HSV slenkstį, vieno kadro analizė „OpenCV“ programos aplinkoje užtruko beveik 2 s, o per 30 pakartojimų neužfiksuota klaidingų teigiamų ar neigiamų rezultatų. Panašius rezultatus pasiekti su specializuotu pramoniniu vaizdo jutikliu būtų bent 6 kartus brangiau, tačiau tik 0,3 s greičiau, todėl internetinės kameros sprendimas pripažintas ekonomiškai naudingesniu.

Koh‘as ir kt. [17] rekomenduoja naudoti „Grad-CAM“, „SmoothGrad“ ir SHAP žemėlapius kaip standartą ekranų pramonėje, kadangi jie leidžia inžinieriui matyti, ar modelis tam tikrą savo sprendimą grindžia realiu defektu, ar foniniu artefaktu. Praktikoje tai leidžia išfiltruoti beveik 3,5 proc. klaidingų pranešimų dar prieš juos perduodant procesoriui.

Klasikinių vaizdo filtrų ir lengvų (angl. *lite*) mokymo metodų sinergiją analizavo autoriai [18] savo apžvalgoje: NDF + GLCM ypatybių vektorius, klasifikuojamas „atsitiktiniu“ algoritmu, tikrinant plonuosius LED sluoksnius, pasiekė 98 proc. tikslumą, o vienai detalei išnagrinėti prireikė tik 0,07–0,11 ms, todėl teoriškai toks metodas leidžia patikrinti apie 14 000 komponentų per sekundę, nekaupiant kadrų.

Aptartų vaizdų atpažinimo metodų rezultatai rodo, kad didžiausias tikslumas gaunamas tada, kai vaizdo analizė suskaidoma į kelis aiškius etapus, o ne bandoma vienu algoritmu išspręsti visą uždavinį. Literatūroje pateikti metodai leidžia pasiekti 91–99 proc. defektų aptikimo tikslumą, 62–125 kadrų per sekundę apdorojimo spartą arba trumpesnę nei 5 ms reakcijos trukmę, tačiau šie rezultatai dažniausiai pasiekiami specializuotose, konkrečiam objektui ir konkrečioms vaizdo gavimo sąlygoms pritaikytose sistemose. Todėl šiame darbe pasirinkta ne universali visų optinių komponentų klasifikavimo schema, o nuoseklus algoritmas, pritaikytas konkrečiam tiriamajam objektui – segmentiniam ekranui ir dviem papildomiems LED ant PCB plokštės. Pirmiausia YOLO modeliu aptinkama PCB ir segmentinio ekrano sritis, po to kadras normalizuojamas pagal nustatytą geometriją, suformuojamos iš anksto apibrėžtos ROI sritys, jose apskaičiuojami šviesumo ir spalviniai požymiai, o galutinis rezultatas pateikiamas kaip kiekvieno segmento ir LED „ON“ / „OFF“ būsena. Tokia struktūra pasirinkta todėl, kad ji leidžia atskirai įvertinti dvi pagrindines

paklaidų grupes: geometrijos ir ROI pozicionavimo paklaidas bei apšvietimo ar atspindžių sukeltas būsenos nustatymo klaidas.

2.6. Skyriaus apibendrinimas

Atlikta vaizdų atpažinimo metodų analizė (žr. 1 lentelę) parodė, kad elektronikos gaminių optinių elementų testavime taikomi tiek klasikiniai vaizdų apdorojimo metodai, tiek giluminio mokymosi modeliai. Klasikiniai metodai, tokie kaip slenkstinis apdorojimas, kraštų aptikimas ar šablonų atitikimas, yra paprasti ir greiti, tačiau jų veikimas stipriai priklauso nuo apšvietimo, objekto padėties kadre ir rankiniu būdu parinktų parametrų. Tiriamojo objekto atveju tai yra esminis trūkumas, nes segmentų ekranas ir du LED indikatoriai gali būti fiksuojami skirtingu kampu, o jų matomumą veikia atspindžiai bei kameros ekspozicija.

1 lentelė. Vaizdų atpažinimo metodų tinkamumo tiriamajam objektui palyginimas

Metodas	**Privalumai**	**Trūkumai**	**Tinkamumas**
Klasikinis slenkstinis apdorojimas	Paprastas, greitas, nereikalauja mokymo duomenų. Tinka, kai apšvietimas ir objekto padėtis stabilūs.	Jautrus apšvietimui, atspindžiams, ekspozicijai ir ROI padėties paklaidoms. Sunku užtikrinti stabilumą keičiantis geometrijai.	Netinkamas kaip pagrindinis metodas, nes tiriamasis objektas gali pasislinkti ar pasisukti kadre, o LED ir segmentų ryškumas priklauso nuo atspindžių.
Šablonų atitikimas	Leidžia lyginti žinomą indikatoriaus struktūrą su kadro fragmentu. Gali būti tikslus, kai objektas fiksuotas vienodoje padėtyje.	Reikalauja pastovios skalės, kampo ir gerai suderinto vaizdo. Netoleruoja didesnių pasukimų, perspektyvos pokyčių ir atspindžių.	Ribotai tinkamas, nes segmentinio ekrano struktūra žinoma, tačiau kamera fiksuoja kintančią PCB padėtį.
CNN klasifikacija	Gali automatiškai išmokti vaizdo požymius ir klasifikuoti sudėtingus atvejus. Literatūroje dažnai pasiekia aukštą tikslumą.	Reikėtų didelio visų galimų segmentų ir LED būsenų duomenų rinkinio. Mažiau aiški klaidų priežastis, sunkiau keisti ROI logiką.	Netinkamas kaip vienintelis sprendimas, nes šiame darbe reikia ne vienos bendros klasės, o konkrečių fizinių segmentų ir LED būsenų.
YOLO aptikimas	Tinka objektams lokalizuoti kintančiomis vaizdo sąlygomis. Leidžia automatiškai rasti PCB ir ekrano sritį.	Vien aptikimas nepasako, kurie konkretūs segmentai ar LED yra įjungti. Reikia papildomos būsenos nustatymo logikos.	Tinkamas PCB ir segmentinio ekrano sričiai aptikti, bet nepakankamas kaip galutinis būsenų nustatymo metodas.
YOLO + ROI	YOLO lokalizuoja objektą, o ROI analizė leidžia vertinti konkrečius segmentus ir LED pagal jų fizinę padėtį. Galima atskirai analizuoti geometrijos ir optinių trikdžių klaidas.	Reikia tiksliai sudaryti ROI išdėstymą ir parinkti būsenų sprendimo slenksčius. Veikimas priklauso nuo ROI pozicionavimo stabilumo.	Pasirinktas metodas, nes geriausiai atitinka darbo tikslą: nustatyti segmentinio ekrano ir LED „ON“ / „OFF“ būsenas iš kameros vaizdo, įvertinant kintančią geometriją.

Giluminio mokymosi modeliai, ypač CNN, YOLO, FCN ir „Transformer“ architektūros, literatūroje pasiekia aukštus aptikimo ir klasifikavimo rezultatus. Analizuotuose moksliniuose šaltiniuose nurodomas 91–99 proc. defektų aptikimo tikslumas, o optimizuoti modeliai gali veikti 62–125 kadrų per sekundę sparta. Tačiau tokie rezultatai dažniausiai gaunami tada, kai sistema sprendžia aiškiai apibrėžtą uždavinį: aptinka defektą, klasifikuoja vaizdą arba segmentuoja objektą. Šiame projekte sprendžiamas kitas uždavinys – reikia ne klasifikuoti visą vaizdą, o nustatyti konkrečių optinių elementų funkcines būsenas, išlaikant ryšį su fizine jų padėtimi PCB plokštėje.

Dėl šios priežasties vien CNN klasifikatoriaus naudojimas nebūtų pakankamai patogus: reikėtų sudaryti didelį visų galimų segmentų ir LED būsenų duomenų rinkinį, o pakeitus tiriamojo objekto geometriją reikėtų papildomai tikslinti modelį. Vien YOLO aptikimas taip pat neišsprendžia uždavinio iki galo, nes jis leidžia nustatyti objekto arba srities padėtį, bet nepateikia kiekvieno segmento ir LED būsenos pagal iš anksto žinomą indikatorių struktūrą. Todėl šiame projekte pasirinktas mišrus sprendimas: YOLO naudojamas PCB ir segmentinio ekrano sričiai aptikti, o galutinė segmentų ir LED būsena nustatoma algoritmiškai, analizuojant iš aptiktos geometrijos suformuotas ROI sritis.

Toks sprendimas leidžia atskirti dvi užduotis: neuroninis tinklas sprendžia lokalizavimo problemą, o ROI analizė – konkrečių indikatorių „ON“ / „OFF“ būsenų nustatymą. Tai svarbu todėl, kad klaidas galima analizuoti atskirai: ar jos susijusios su objekto geometrijos nustatymu, ar su apšvietimu, atspindžiais ir požymių skaičiavimu ROI srityje. Atsižvelgiant į tyrimų objektą ir eksperimentinės sistemos paskirtį, YOLO + ROI metodas pasirinktas kaip praktiškiausias, nes jis nereikalauja klasifikuoti visų galimų būsenų kaip atskirų klasių, bet leidžia išlaikyti aiškų ryšį tarp vaizde aptiktos srities ir fizinių tiriamojo objekto elementų.

1 lentelėje metodai vertinami ne pagal bendrą technologinį naujumą, o pagal jų praktinį tinkamumą šio darbo sistemai. Svarbiausi kriterijai yra šie: ar metodas gali lokalizuoti PCB ir ekraną kintančioje geometrijoje, kiek mokymo duomenų jam reikia, ar sprendimas išlieka interpretuojamas, ar metodas jautrus apšvietimo pokyčiams ir ar galima tiesiogiai gauti kiekvieno segmento bei LED būseną. Remiantis šiuo palyginimu, tolesniuose projekto skyriuose taikomas YOLO + ROI metodas, nes jis leidžia suderinti automatinį objekto lokalizavimą su aiškia ir koreguojama indikatorių būsenų nustatymo logika.

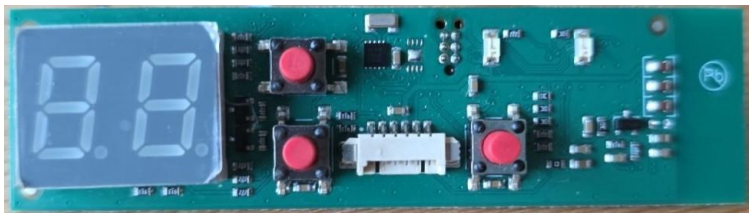
3. Tiriamasis objektas ir eksperimentinė sistema

Šiame skyriuje aiškiai atskiriami trys tyrimo lygmenys. Pirmasis lygmuo yra fizinis tiriamasis objektas – PCB su dviejų skaitmenų segmentiniu ekranu ir dviem papildomais LED. Antrasis lygmuo yra valdymo ir duomenų rinkimo sistema – ESP32-S3 valdiklis, USB kamera ir kompiuterio programa. Trečiasis lygmuo yra matuojami rezultatų kintamieji – 14 segmentų „ON“ / „OFF“ būsenos, dviejų LED būsenos, kadrų lygio tikslumas ir klaidų tipai.

Eksperimentinėje sistemoje mikrovaldiklis generuoja žinomas etalonines būsenas, o kamera fiksuoja jų vaizdus, todėl kiekvienam kadrui galima priskirti teisingą atsakymą be rankinio būsenų žymėjimo. Pirminiame duomenų rinkimo plane numatyta 456 būsenų kombinacijų seka: dviejų skaitmenų rodmenys nuo „00“ iki „99“ su keturiomis papildomų LED kombinacijomis ir 14 atskirų segmentų būsenų, taip pat su keturiomis LED kombinacijomis. Kiekviena kombinacija kartojama 15 kartų, keičiant objekto padėtį kameros atžvilgiu ir, kai taikoma, fiksavimo apšvietimo sąlygas. Vėlesniame tyrimo etape galutiniam vertinimui naudota ne visa pirminė seka, o atrinkta 840 kadrų imtis, orientuota į atskirų segmentų ir LED būsenų atpažinimą.

3.1. Tiriamasis objektas

Šiame projekte tiriamasis objektas (žr. 12 pav.) – tai spausdintinė plokštė (PCB) su šviesos diodų indikacija, kurią sudaro segmentų ekranas (DS200) ir du papildomi šviesos diodai (toliau – papildomi LED). Tiriamasis mazgas parinktas todėl, kad jo optinė būsena yra aiškiai apibrėžiama, valdomai keičiama ir gali būti automatiškai įvertinama iš vaizdo kadrų, remiantis segmentų ir LED švytėjimo požymiais. Tokia struktūra leidžia formuoti valdomą būsenų rinkinį ir sistemiškai tirti vaizdų apdorojimo metodo patikimumą, kai kinta fiksavimo geometrija ir aplinkos apšvietimo sąlygos.



12 pav. Tiriamasis objektas

Tiriamojo objekto išėjimas šiame tyrime suprantamas kaip optinė būsena, t. y. vienu metu įjungtų / išjungtų šviesos elementų kombinacija:

- 2 × 7 segmentų ekrano segmentų įjungimo / išjungimo būsena;
- dviejų papildomų LED įjungimo / išjungimo būsena.

Kadangi eksperimentinėje sistemoje optinės būsenos generuojamos ne rankiniu būdu, o mikrovaldikliu, kiekvienam užfiksuotam kadrui gali būti priskiriama etaloninė būsena. Ši etaloninė būsena gaunama iš ESP32-S3 valdiklio siunčiamo ar vykdomo būsenos kodo: skaitmenų rodymo režimu nustatoma dviejų skaitmenų reikšmė, o atskirų segmentų tikrinimo režimu – konkretus aktyvus segmentas. Papildomų LED būsena nustatoma pagal atskirą LED režimo kodą, kuris apibrėžia keturias galimas kombinacijas: abu LED išjungti, įjungtas tik LED1, įjungti abu LED arba įjungtas tik LED2. Taip vaizdo analizės algoritmo rezultatas gali būti lyginamas ne su rankiniu

stebėjimu, o su valdiklio sugeneruota būsena. Etaloninių būsenų sudarymo logika pateikiama 2 lentelėje.

2 lentelė. Etaloninių būsenų sudarymas iš ESP32-S3 valdiklio komandų

Valdiklio būsena / režimas	**Reikšmė eksperimente**	**Etaloninė būsena vaizdo analizei**	**Naudojimas vertinime**
Skaitmenų rodymo režimas	Generuojamos reikšmės nuo „00“ iki „99“	Pagal rodomą skaičių žinoma, kurie 7 segmentų ekrano segmentai turi būti įjungti kiekviename iš dviejų skaitmenų	Naudojama tikrinti, ar algoritmas teisingai nustato visų segmentų „ON“ / „OFF“ būsenas
Atskirų segmentų režimas	Generuojamos 14 atskirų segmentų būsenų: A1–G1 ir A2–G2	Vienu metu aktyvus vienas konkretus segmentas, kiti segmentai laikomi išjungtais	Naudojama tikrinti ROI pozicionavimą ir pavienių segmentų atpažinimą
LED režimas 00	Abu papildomi LED išjungti	LED1 = „OFF“, LED2 = „OFF“	Naudojama kaip kontrolinė būsena be papildomo LED švytėjimo
LED režimas 10	Įjungtas LED1	LED1 = „ON“, LED2 = „OFF“	Naudojama LED1 atpažinimui tikrinti
LED režimas 11	Įjungti abu LED	LED1 = „ON“, LED2 = „ON“	Naudojama abiejų LED vienalaikio atpažinimo patikrai
LED režimas 01	Įjungtas LED2	LED1 = „OFF“, LED2 = „ON“	Naudojama LED2 atpažinimui tikrinti

Sudaryta etaloninė būsenų struktūra leidžia kiekvieną vaizdo kadrą vertinti kiekybiškai. Algoritmo nustatytos segmentų ir LED būsenos lyginamos su valdiklio sugeneruotomis būsenomis, todėl galima apskaičiuoti kadrų atpažinimo tikslumą, segmentų būsenų tikslumą, LED būsenų tikslumą ir nustatyti klaidingų atpažinimų priežastis.

Taigi tyrimo uždavinys optinių duomenų lygmenyje yra ne „pamatyti skaičių“, o iš vaizdo kadro patikimai nustatyti, kurie segmentai ir kurie papildomi LED tuo momentu yra įjungti. Šis pasirinkimas yra svarbus, nes segmentų ekranas yra sudarytas iš atskirų šviesos elementų, kurių šviesumas ir kontūrai vaizde gali kisti dėl aplinkos apšvietimo, atspindžių, kameros žiūrėjimo kampo ir tiriamojo objekto padėties pokyčių. Dėl to realiame fiksavime skaitmens „00“ ar „99“ interpretacija tampa išvestiniu rezultatu, o pirminė klasifikavimo užduotis yra segmentų būsenų atpažinimas.

Papildomi LED šiame projekte traktuojami kaip atskira būsenos dalis, nes jų švytėjimas:

- suteikia papildomą informacijos kanalą (papildomas binarinis požymis);
- leidžia patikrinti, ar algoritmas vienodai patikimai identifikuoja skirtingo dydžio ir formos šviesos šaltinius tame pačiame kadre;
- gali būti naudojamas kaip kontrolinis indikatorius (pvz., būsenos perjungimui patvirtinti), jeigu tai numatyta bandymų sekoje.

Tiriamajame objekte išskiriamos dvi elementų grupės (žr. 3 lentelę): optiniai elementai, kurių būsena yra vertinama kaip algoritmo rezultatas, ir geometriniai PCB bruožai, reikalingi tiriamojo objekto padėčiai vaizde nustatyti. Optiniams elementams priskiriamas segmentų ekranas ir du papildomi LED indikatoriai. Šių elementų būsena vertinama binariškai, t. y. kiekvienas segmentas arba LED laikomas įjungtu arba išjungtu. Geometriniai PCB bruožai nėra vertinami kaip rezultatas, tačiau jie svarbūs lokalizuojant objektą kadre ir kompensuojant padėties pokyčius, kai PCB kameros atžvilgiu pasislenka, pasisuka arba keičiasi jos mastelis vaizde.

3 lentelė. Tiriamojo objekto elementai ir jų vaidmuo tyrime

Elementas	Trumpas aprašymas	Vaidmuo tyrime
Segmentų ekranas	Du skaitmenys, sudaryti iš A–G segmentų	Pagrindinis atpažįstamas objektas. Vertinamos 14 segmentų „ON“ / „OFF“ būsenos: A1–G1 ir A2–G2
Papildomas LED 1	Atskirai valdomas LED indikatorius	Vertinama viena binarinė būsena: LED 1 įjungtas arba išjungtas
Papildomas LED 2	Atskirai valdomas LED indikatorius	Vertinama viena binarinė būsena: LED 2 įjungtas arba išjungtas
PCB sritis	Plokštė, ant kurios sumontuotas ekranas ir LED	Naudojama tiriamojo objekto lokalizavimui kadre ir geometrijos pokyčiams įvertinti
Segmentinio ekrano sritis	Ekrano vieta PCB plokštėje	Naudojama ROI sričių išdėstymui susieti su realia indikatoriaus padėtimi
PCB kontūro bruožai	Plokštės kraštai, forma ir kiti stabilūs geometriniai požymiai	Gali būti naudojami kaip papildoma informacija objekto padėčiai, pasukimui ir masteliui įvertinti

Taip atskyrus tiriamojo objekto elementus, aiškiai apibrėžiama, kurie komponentai sudaro algoritmo išvestį, o kurie naudojami tik vaizdo geometrijai nustatyti. Tolesniuose projekto skyriuose segmentų ir LED būsenos lyginamos su valdiklio sugeneruotomis etaloninėmis būsenomis, o PCB ir ekrano sritys naudojamos ROI pozicionavimui bei kintančios geometrijos įtakai mažinti.

3.2. Naudojama įranga ir jos parametrai

Eksperimentinė sistema (žr. 13 pav.) sudaryta iš vaizdo fiksavimo, tiriamojo objekto būsenų valdymo posistemių ir valdymo programos kompiuteryje. Šios sistemos paskirtis – automatiškai generuoti tiriamojo objekto optines būsenas ir kiekvienai būsenai fiksuoti vaizdą, kurį būtų galima apdoroti.

13 pav. Eksperimentinė sistema

Sistema realizuojama naudojant kompiuterį „MSI Thin“ 15 B13VE su „Python“ programine įranga, kuri inicijuoja bandymo ciklą ir vykdo būsenų perjungimą. Būsenos perjungiamos cikliškai, o vaizdo kadrai fiksuojami sinchronizuotai su būsenų seka. Kadangi tyrime numatyta kintanti fiksavimo geometrija, vaizdų analizės dalyje negalima remtis fiksuotomis ROI koordinatėmis, todėl šiame poskyryje akcentuojami tie aparatūros parametrai, kurie tiesiogiai veikia vaizdo kokybę ir kadrų palyginamumą. Kompiuterio techninės charakteristikos pateiktos 4 lentelėje.

4 lentelė. Kompiuterio „MSI Thin“ 15 B13VE techninės charakteristikos [41]

Parametras	**Reikšmė**
Procesorius (CPU)	Iki 13-os kartos „Intel® Core™“ i7 procesoriaus
Operacinė sistema (OS)	„Windows“ 11 Home (MSI rekomenduoja „Windows“ 11 Pro verslo naudojimui); „Windows“ 11 Pro
Ekranas	15,6" FHD (1920×1080), 144 Hz, IPS lygio
Vaizdo plokštė (GPU)	„NVIDIA® GeForce RTX™“ 4050 nešiojamojo kompiuterio GPU, 6 GB GDDR6; pažangiam DI iki 194 AI TOPS
GPU dažnis / galia	1605 MHz (iki „Boost“); 45 W maksimali vaizdo plokštės galia su „Dynamic Boost“ (gali skirtis priklausomai nuo pasirinktų specifikacijų)
Operatyvioji atmintis (RAM)	DDR4-3200; maks. 64 GB; 2 lizdai
Saugyklos galimybės	1 × M.2 SSD lizdas (NVMe PCIe Gen4); 1× 2,5" SATA HDD; (SSD atnaujinimui kreiptis į serviso centrą)
Saugumas	Patikimasis programinės aparatinės įrangos platformos modulis fTPM 2.0; „Kensington“ užraktas
Internetinė kamera	HD tipo (30 k./s @ 720p)
Klaviatūra	Vienos spalvos apšvietimas (mėlynas)
Ryšiai	Gigabitinis LAN; 802.11ax Wi-Fi 6E + Bluetooth v5.3
Garsas	2 × 2 W garsiakalbiai
Garso jungtys	1 × mikrofono įvestis, 1× ausinių išvestis
Prievadai (I/O)	1 × RJ45; 1 × Type-C (USB 3.2 Gen1 / DisplayPort™); 3× Type-A USB 3.2 Gen1; 1× HDMI™ (4K @ 30 Hz)
Baterija	3 elementų, 52,4 Wh

Vaizdui fiksuoti naudojama „Logitech QuickCam Communicate MP“ USB kamera (žr. 14 pav.) su fiksuotu fokusavimu. Pagrindinės kameros charakteristikos pateiktos 5 lentelėje.

Kameros vaizdo režimas parenkamas taip, kad būtų užtikrintas pakankamas segmentų formos atskyrimas ir stabilus kadrų gavimas realiuoju laiku. Tyrime taikomi šie fiksavimo nustatymai:

- raiška – 640 × 480 taškų;
- kadrų dažnis – 30 k./s;
- fokusavimas – fiksuotas;
- atstumas iki PCB – apie 20 cm;
- apšvietimas – aplinkos (angl. *ambient*), be specialaus papildomo šviestuvo.

Kameros ekspozicija, baltos spalvos balansas ir signalo stiprinimas tyrimo metu nebuvo fiksuojami aparatūriškai – naudoti kameros automatiniai nustatymai. Šis sprendimas leido įvertinti algoritmą praktiškesnėmis sąlygomis, tačiau kartu tai yra tyrimo apribojimas: automatinis ekspozicijos ir baltos spalvos balanso prisitaikymas galėjo keisti segmentų bei LED ryškumo pasiskirstymą kadre. Dėl šios

priežasties vėliau klaidų analizėje atskirai aptariama atspindžių, lokalaus ryškumo ir ekspozicijos prisitaikymo įtaka.

14 pav. „Logitech QuickCam Communicate MP“ USB kamera

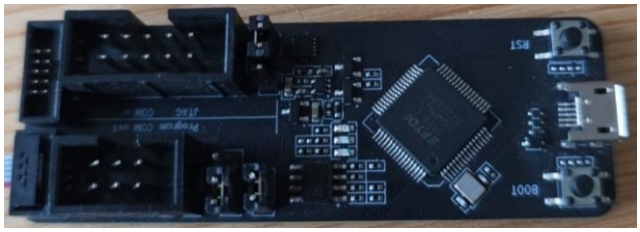
5 lentelė. Pagrindinės „Logitech QuickCam Communicate MP“ USB kameros charakteristikos [42]

Parametras	**Reikšmė**
Vaizdo jutiklis	1 280 × 960 jutiklis
Fokusavimas	Fiksuotas fokusavimas
Efektyvi skiriamoji geba	4.0 megapikseliai, su įrašytais programinės įrangos patobulinimais
Nuotraukos skiriamoji geba	1 280 × 960
Vaizdo įrašo skiriamoji geba	1 280 × 960
Kadrų skaičius per sekundę	30 kps
Garsas	Integruotas mikrofonas
Prievadas	USB 2.0 (USB 1.1 palaikymas)
Maitinimo reikalavimai	Maitinimas per USB jungtį

Tiriamojo objekto padėtis kameros atžvilgiu nėra griežtai fiksuota, todėl matavimo eigoje galimi plokštės poslinkiai ir pasukimai kadro plokštumoje bei nedideli perspektyviniai pokyčiai. Šis aspektas yra sąmoningai paliekamas kaip tyrimo sąlyga, nes taip surenkami duomenys geriau atspindi realias naudojimo situacijas, kai objektas nėra įstatomas į preciziškai vienodą padėtį kiekvieno fiksavimo metu. Būsenoms generuoti naudojamas mikrovaldiklis „ESP32-S3“ (žr. 15 pav. ir 6 lentelę), kuris valdo segmentų ekraną ir papildomus LED. Mikrovaldiklio paskirtis šiame tyrime – užtikrinti vienodą

ir pakartojamą būsenų perjungimą, sudaryti sąlygas automatiškai kartoti bandymus su identiška būsenų seka. Programavimo ir derinimo tikslais naudojamas atskiras programavimo / derinimo įrenginys „ESP-Prog“ (žr. 16 pav. ir 7 lentelę), leidžiantis įkelti programą į mikrovaldiklį ir patikrinti ryšio veikimą. Ši aparatūros dalis nėra tiriama kaip atskiras objektas, tačiau ji būtina užtikrinti bandymų pakartojamumą ir stabilų sistemos paruošimą duomenims rinkti.

15 pav. „ESP32-S3“ mikrovaldiklis



16 pav. „ESP-Prog“ programatorius

6 lentelė. Pagrindinės ESP32-S3 mikrovaldiklio charakteristikos [43]

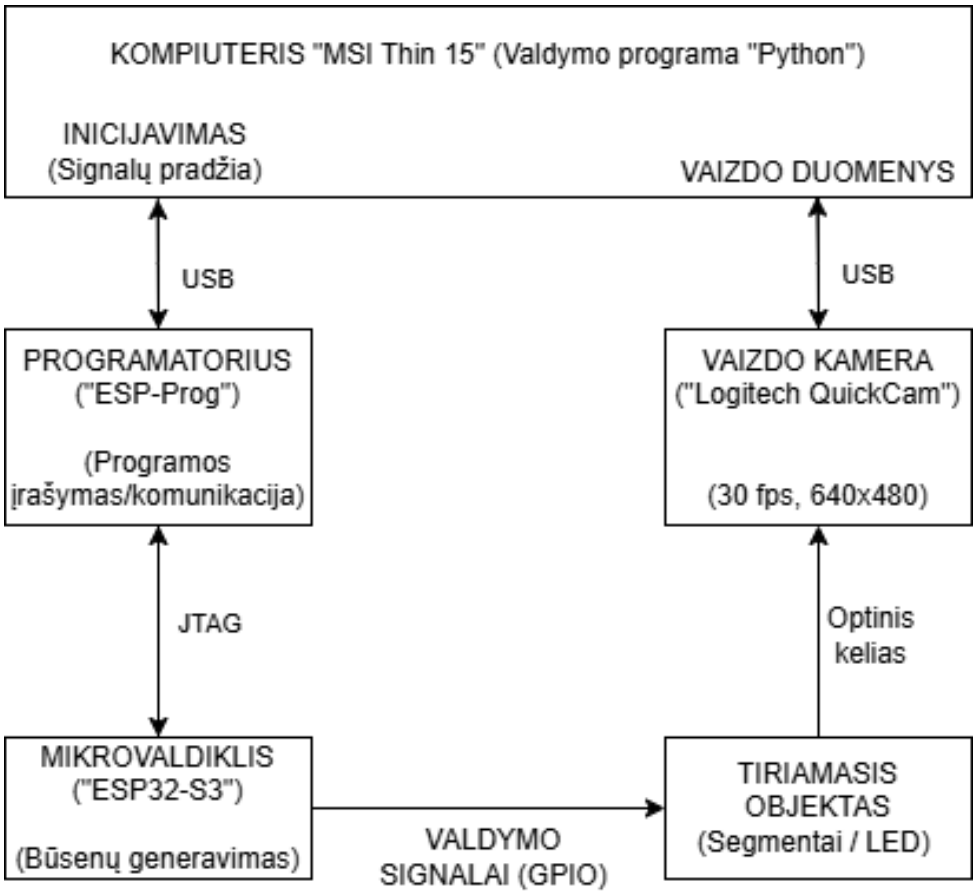
Charakteristika	**Reikšmė**
Procesorius	Dviejų branduolių 32 bitų LX7, iki 240 MHz
Atmintis	384 KB nuolatinė + 512 KB operatyvioji + 16 KB laikrodžio srities
Programuojami įvadai/išvadai	Iki 45 programuojamų įvadų/išvadų
Nuosekliosios sąsajos	3 nuosekliosios sąsajos
Universali nuoseklioji jungtis	Veikia kaip kompiuterio įrenginys ir kaip pagrindinis įrenginys
Laikmačiai	4 bendros paskirties + sistemos laikmatis
Impulsų pločio moduliavimas	Iki 8 kanalų šviesos diodams
Belaidis tinklas ir mažos energijos ryšys	IEEE 802.11 b/g/n (2,4 GHz, iki 150 Mb/s) + 5 versija (iki 20 dBm)

7 lentelė. Pagrindinės „ESP-Prog“ programatoriaus charakteristikos [44]

Parametras	**Reikšmė**
Įrenginio tipas	Programavimo ir derinimo įrankis
Pagrindinės paskirtys	Programinės aparatinės įrangos įkėlimas; nuoseklusis ryšys; derinimas per JTAG
Įkėlimo ir nuosekliojo ryšio palaikomi valdikliai	ESP8266, ESP32, ESP32-S2, ESP32-S3, ESP32-C3
JTAG derinimo palaikomi valdikliai	ESP32, ESP32-S2, ESP32-S3, ESP32-C3
Prijungimas prie kompiuterio	Vienas universalus nuoseklusis kabelis; kompiuteris atpažįsta programavimo ir JTAG sąsajas pagal prievadų numerius
Maitinimo įtampa iš įrankio (per kontaktų lizdus)	5 V arba 3,3 V (pasirenkama)
Signalų lygis	Priėmimo / siuntimo ir JTAG signalai visada 3,3 V lygio

17 paveiksle pateikiama eksperimentinės sistemos blokinė schema, kurioje parodyti pagrindiniai valdymo ir duomenų srautai.

Joje matyti, kaip valdymo programa inicijuoja būsenų generavimą mikrovaldiklyje, o vaizdo kamera perduoda fiksuotus kadrus į kompiuterį tolesnei analizei.



17 pav. Blokinė sistemos schema

Pateikta blokinė schema apibrėžia pagrindines signalų sąsajas tarp posistemių ir parodo, kad vaizdo fiksavimo bei būsenų valdymo grandys yra valdomos iš kompiuterio, o mikrovaldiklis užtikrina pakartojamą tiriamojo objekto optinių būsenų generavimą.

3.3. Bandymų planas ir duomenų rinkimo seka

Šiame poskyryje apibrėžiamas bandymų planas ir duomenų rinkimo eiga. Esminė idėja – automatiškai perjungti segmentų ekrano ir papildomas LED būsenas ciklu, o kiekvienai būsenai užfiksuoti vaizdo kameros kadrą. Taip suformuojamas pažymėtas duomenų rinkinys, tinkamas 4 skyriuje aprašomiems vaizdų apdorojimo algoritmams vertinti.

Būsenų seka sudaryta taip, kad apimtų tiek dviejų skaitmenų rodmenis, tiek segmentų būsenas. Svarbu, kad tai nėra atskiri režimai – visos būsenos yra vienoje bendroje ciklinėje sekoje. Bendrai užfiksuojamos 456 kombinacijos:

- dviejų skaitmenų rodmenys nuo „00“ iki „99“, kiekvienam rodmeniui taikant keturias papildomas LED būsenas;
- 14 segmentų būsenas (kai įjungiamas konkretus segmentas);

– kiekvienai būsenai taip pat taikant keturias, dviejų papildomų LED būsenas;
– taškų (DP) ir „visiškai išjungta“ būsenos į bandymų rinkinį neįtraukiamos (netestuojamos).

Pradiniame tyrimo etape bandymų seka buvo sudaryta plačiau, įtraukiant ne tik pavienių segmentų būsenas, bet ir skaitines dviejų segmentų ekrano reikšmes nuo „00“ iki „99“. Tokia seka leido surinkti platesnį vaizdų rinkinį ir įvertinti, kaip kamera fiksuoja skirtingas ekrano kombinacijas. Vis dėlto vėlesniame tyrimo etape skaitinių reikšmių kaip atskirų simbolių atpažinimo atsisakyta, nes darbo tikslas buvo susiaurintas iki optinių elementų funkcinių būsenų nustatymo, t. y. atskirų segmentų ir LED indikatorių „ON“ / „OFF“ būsenų atpažinimo.

Duomenų rinkimo metu kiekvienai kombinacijai fiksuojamas vienas kadras, tačiau siekiant įvertinti algoritmo atsparumą geometrijos ir aplinkos pokyčiams, kombinacijos buvo kartojamos keičiant tiriamojo objekto padėtį kameros atžvilgiu ir fiksuojant kadrus skirtingomis aplinkos apšvietimo sąlygomis. Pradiniame duomenų rinkimo etape, įtraukus skaitines ekrano reikšmes, buvo surinktas 6 840 kadrų rinkinys. Šis rinkinys naudotas sistemos veikimui patikrinti ir vaizdų apdorojimo algoritmo kūrimo eigai, tačiau galutiniame eksperimentiniame vertinime skaitinių reikšmių atpažinimas nebebuvo analizuojamas. Todėl galutiniam algoritmo tikslumo vertinimui naudota atskira 840 kadrų imtis, sudaryta iš pavienių segmentų ir LED būsenų kombinacijų.

Kombinacijų perjungimas ir kadrų fiksavimas vykdomas automatizuotai. Vienai būsenai numatytos dvi laiko pauzės:
– 200 ms pauzė po būsenos nustatymo, kad šviesos diodai nusistovėtų;
– 400 ms pauzė po nusistovėjimo (iki kitos būsenos), užtikrinanti pakankamą laiko tarpą stabiliai būsenų kaitai ir duomenų įrašymui.

Kad būtų aišku, kada tiksliai fiksuojamas kadras, būsenos laiko seka pateikiama diagramoje (žr. 18 pav.), kuri parodo, kad kadras fiksuojamas ne iškart po perjungimo, o po nusistovėjimo pauzės. Tai yra svarbu, nes priešingu atveju algoritmas būtų maitinamas „pereinamaisiais“ kadrais, kurie neturi stabilios optinės būsenos.

18 pav. Būsenos laiko sekos diagrama

Duomenų rinkimo eiga atliekama padedant tiriamąjį objektą apytiksliai į kameros matymo lauką, o bandymo ciklas paleidžiamas kompiuterio programoje paspaudus „START“. Toliau sistema pati ciklišk ai perjungia kombinacijas ir fiksuoja kadrus. Kadangi būseną tuo metu generuoja mikrovaldiklis, etaloninė (teisinga) būsena yra žinoma iš valdymo sekos ir gali būti priskiriama kiekvienam užfiksuotam kadrui. Tokia duomenų žymėjimo logika yra patogi tuo, kad nereikia rankinio anotavimo – etalonas gaunamas automatiškai iš bandymo eigos. Taip pradiniame etape buvo sudarytas 6 840 kadrų rinkinys, tačiau galutiniame tyrimo etape jis nebuvo naudojamas visas rezultatų statistikai skaičiuoti, nes atsisakyta skaitinių ekrano reikšmių analizės ir pereita prie atskirų segmentų bei LED indikatorių būsenų vertinimo.

3.4. Skyriaus apibendrinimas

Aprašyta eksperimentinė sistema leidžia valdomai generuoti segmentinio ekrano ir dviejų papildomų LED būsenas bei kiekvienai sugeneruotai būsenai užfiksuoti vaizdo kadrą. Būsenas generuoja ESP32-S3 valdiklis, todėl kiekvienam kadrui gali būti priskiriama etaloninė būsena be papildomo rankinio žymėjimo. Sistemoje naudojamas segmentų ekranas ir du papildomi LED indikatoriai, o bandymų seka sudaroma iš skaitmeninių reikšmių, atskirų segmentų būsenų ir keturių LED kombinacijų. Kadangi PCB padėtis kameros atžvilgiu nėra pastovi, surinkti kadrai leidžia vertinti ne tik segmentų ir LED būsenų atpažinimą, bet ir algoritmo atsparumą padėties, pasukimo, apšvietimo bei atspindžių pokyčiams. Šiame skyriuje apibrėžti tiriamojo objekto elementai, naudojamos įrangos charakteristikos, vaizdo fiksavimo sąlygos ir būsenų generavimo logika, kuri vėliau naudojama algoritmui kurti ir jo tikslumui eksperimentiškai įvertinti. 6 840 kadrų rinkinys šiame darbe laikomas pirminiu duomenų rinkimo etapu. Jame buvo įtrauktos ir skaitinės segmentų ekrano reikšmės, tačiau galutinis tyrimas orientuotas ne į skaitmens kaip simbolio atpažinimą, o į pavienių segmentų ir LED indikatorių funkcinių būsenų nustatymą.

4. Vaizdų apdorojimo metodika ir algoritmai

Šiame skyriuje aprašoma vaizdų apdorojimo metodika, skirta iš kameros kadro nustatyti segmentinio ekrano ir dviejų papildomų LED optines būsenas. Kadangi tiriamojo objekto padėtis kameros atžvilgiu nėra pastovi, algoritmas pirmiausia turi nustatyti PCB ir ekrano srities padėtį, o tik tada taikyti ROI pagrindu veikiančią būsenų analizę. Taip pat šiame skyriuje pristatoma visa apdorojimo grandinė: kadro gavimas, YOLOv5 aptikimas, geometrijos suvienodinimas, ROI sudarymas, požymių skaičiavimas, „ON“ / „OFF“ (įjungta / išjungta) sprendimo priėmimas ir klaidų fiksavimas.

Pakartojamumui šiame skyriuje pateikiama ne tik bendra schema, bet ir esminiai algoritminiai parametrai: YOLOv5 aptikimo logika, geometrijos suvienodinimo principas, ROI sudarymo taisyklės, požymių skaičiavimo formulės, slenksčių parinkimo tvarka ir klaidų kodai. Galutinė 840 kadrų vertinimo imtis naudojama rezultatams skaičiuoti, o ne metodo slenksčiams priderinti.

Kadangi šiame projekte sprendžiamas ne skaitmens kaip simbolio atpažinimo, o atskirų optinių elementų būsenų nustatymo uždavinys, vaizdo apdorojimo algoritmas sudarytas iš kelių nuoseklių etapų (žr. 19 pav.). Pirmiausia iš kameros gaunamas kadras, kuriame tiriamasis objektas gali būti pasislinkęs, pasuktas arba nevienodai apšviestas. Toliau YOLOv5 modeliu aptinkama PCB ir segmentinio ekrano sritis. Ši informacija naudojama kadro geometrijai suvienodinti ir ROI sritims į realią indikatorių padėtį kadre perkelti. Tik po šio etapo skaičiuojami segmentų ir LED šviesumo bei spalviniai požymiai, pagal kuriuos priimamas „ON“ / „OFF“ būsenos sprendimas. Algoritmo rezultatas lyginamas su valdiklio sugeneruota etalonine būsena, todėl galima fiksuoti ne tik teisingus atpažinimus, bet ir klaidų tipą.

19 pav. Vaizdų apdorojimo algoritmo blokinė schema

Šioje schemoje (žr. 19 pav.) atskiriami du pagrindiniai algoritmo uždaviniai. Pirmoji dalis skirta objekto padėčiai vaizde nustatyti: tam naudojamas YOLOv5 aptikimas ir geometrijos suvienodinimas. Antroji dalis skirta funkcinei būsenai įvertinti: pagal suformuotas ROI sritis apskaičiuojami požymiai ir nustatoma, kurie segmentai bei LED yra įjungti. Toks išskaidymas reikalingas todėl, kad klaidos gali atsirasti skirtinguose etapuose. Jei netiksliai aptinkama PCB ar ekrano sritis, ROI gali nepataikyti į tikruosius segmentus. Jei geometrija nustatyta teisingai, bet yra atspindžių arba nevienodas apšvietimas, klaida gali atsirasti jau būsenos sprendimo etape. Dėl to rezultatų analizėje atskirai vertinami bendras kadrų tikslumas, segmentų būsenų tikslumas, LED būsenų tikslumas ir klaidingų atpažinimų priežastys.

4.1. Bendras vaizdų apdorojimo planas

Vaizdų apdorojimo eiga šiame projekte sudaryta taip, kad kiekvienam užfiksuotam kadrui būtų galima priskirti vienareikšmį rezultatą – segmentų ir LED būsenų rinkinį. Kadangi geometrija kinta, pirmasis žingsnis nėra ROI analizė, o vaizdo registracija į bendrą koordinačių sistemą. Tik po registracijos galima patikimai taikyti tas pačias ROI taisykles visiems kadrams.

Algoritmo įėjimai:

- kameros kadras I_i, kuriame matomas tiriamasis objektas;
- apmokytas YOLOv5 segmentavimo modelis, aptinkantis PCB ir segmentinio ekrano sritis;

YOLOv5 modelis šiame darbe naudojamas tik dviem lokalizavimo klasėms – PCB ir segmentinio ekrano sričiai – aptikti. Galutinis segmentų arba LED būsenų sprendimas nėra perduodamas DNT klasifikatoriui; jis priimamas pagal ROI srityse apskaičiuotus požymius. Taip užtikrinama, kad mokymo duomenų poreikis apsiribotų objekto lokalizavimu, o galutinis būsenų sprendimas liktų interpretuojamas ir koreguojamas.

- etaloninis ROI išdėstymo failas, kuriame apibrėžtos segmentų ir LED sritys normalizuotoje koordinačių sistemoje;
- valdiklio sugeneruota etaloninė būsena G_i, pagal kurią žinoma, kurie segmentai ir LED turi būti įjungti.

Algoritmo išėjimai:

- 14 segmentų būsenų vektorius S_i = [A1, B1, C1, D1, E1, F1, G1, A2, B2, C2, D2, E2, F2, G2], kur kiekviena reikšmė yra 1 (įjungta) arba 0 (išjungta);
- LED būsenų vektorius L_i = [LED1, LED2];
- kadro įvertinimas: teisingai atpažintas / klaidingai atpažintas;
- klaidos tipas, jeigu rezultatas nesutampa su etalonine būsena.

Bendras vaizdų apdorojimo algoritmas

1. Nuskaitomas kameros kadras I_i.
2. Kadras pateikiamas YOLOv5 segmentavimo modeliui.
3. Tikrinama, ar kadre aptiktos abi būtinos sritys: PCB ir segmentinio ekrano sritis. Jeigu bent viena iš šių sričių neaptikta, kadras žymimas kaip netinkamas geometrijai nustatyti ir registruojama aptikimo klaida.
4. Iš YOLO kaukių apskaičiuojami PCB ir ekrano srities kontūrai. Pagal kontūrus nustatomi jų ribiniai keturkampiai.

5. Apskaičiuojama transformacija iš etaloninės koordinačių sistemos į einamąjį kadrą. Transformacija parenkama taip, kad etaloninė ekrano sritis kuo geriau sutaptų su YOLO modeliu aptikta ekrano sritimi.
6. Pagal apskaičiuotą transformaciją į einamąjį kadrą perkeliamos visos ROI sritys: 14 segmentų ROI, LED1 ROI ir LED2 ROI.
7. Kiekvienoje segmentų ROI srityje apskaičiuojami šviesumo požymiai. Kiekvienoje LED ROI srityje apskaičiuojami šviesumo ir spalviniai požymiai.
8. Kiekvienam segmentui priimamas binarinis sprendimas: jei ROI požymiai viršija nustatytus slenksčius, segmentas laikomas įjungtu; jei požymiai slenksčių neviršija, segmentas laikomas išjungtu.
9. Kiekvienam LED priimamas binarinis sprendimas pagal jo spalvinį ir ryškumo požymį: LED1 vertinamas pagal raudonos šviesos požymius; LED2 vertinamas pagal mėlynos šviesos požymius.
10. Gautas segmentų ir LED būsenų rinkinys palyginamas su valdiklio sugeneruota etalonine būsena G_i.
11. Jeigu visos segmentų ir LED būsenos sutampa su etalonu, kadras laikomas atpažintu teisingai.
12. Jeigu bent viena būsena nesutampa, kadras laikomas klaidingu. Klaida priskiriama vienai iš grupių: geometrijos / ROI padėties klaida, segmento būsenos klaida arba LED būsenos klaida.

Toks algoritminis išskaidymas leidžia atskirai vertinti lokalizavimo ir būsenos nustatymo etapus. PCB ir ekrano srities aptikimas reikalingas tam, kad ROI būtų perkelti į teisingą kadro vietą. Požymių skaičiavimas ROI srityse naudojamas jau ne objekto paieškai, o konkrečiai optinei būsenai nustatyti. Todėl klaidos analizėje galima atskirti, ar neteisingas rezultatas atsirado dėl geometrijos nustatymo, ar dėl šviesumo, spalvos ir atspindžių įtakos ROI požymiams.

4.2. Geometrijos suvienodinimas pagal PCB ir ekraną

Kadangi tiriamojo objekto padėtis kameros matymo lauke nėra fiksuota, skirtingi kadrai tarpusavyje skiriasi poslinkiu, pasukimu ir perspektyviniais iškraipymais. Jeigu šie pokyčiai nebūtų kompensuojami, ROI sritys neapimtų segmentų ir LED, o sprendimas taptų atsitiktinis. Dėl to šiame projekte pasirenkamas geometrijos suvienodinimas remiantis PCB plokštės geometrija ir segmentų ekranu, nes tai yra stabilūs ir aiškiai apibrėžti geometriniai orientyrai.

Registracijos logika paremta tuo, kad plokštės forma ir ekrano padėtis:
- yra pasikartojantis, geometriniu požiūriu stabilus elementas;
- paprastai turi aiškų kontrastą ir keturkampę formą, todėl yra tinkami automatinei paieškai;
- leidžia atkurti PCB padėtį net tada, kai ekrano segmentai ar papildomi LED trukdo detekcijai.

Suvienodinimas atliekamas etapais. Pirmiausia YOLOv5 modelis aptinka PCB ir segmentinio ekrano sritis. Iš segmentavimo kaukių apskaičiuojami kontūrai ir ribiniai keturkampiai, o ekrano ir PCB tarpusavio padėtis naudojama orientacijai nustatyti. Galiausiai apskaičiuojama perspektyvinė transformacija, kuri etaloninį ROI išdėstymą perkelia į einamąjį kadrą. Taip tiriamojo objekto padėtis kiekviename kadre susiejama su ta pačia normalizuota koordinačių sistema.

Svarbu, kad toks registracijos metodas leidžia suvaldyti ir nepatogius atvejus, kai plokštė kadre pasukta ar apsukta. Apsukimas nėra atskiras režimas – tai tiesiog kita to paties geometrinio ryšio realizacija, kurią galima atpažinti iš ekrano ir PCB tarpusavio išsidėstymo (t. y. iš to, kur ekranas, kurioje PCB vietoje atsiduria po pasukimo).

Kad registracija būtų patikima, tikrinama, ar aptiktas ekranas ir PCB atitinka tikėtinus formos, dydžio ir tarpusavio padėties santykius. Praktikoje tai reiškia, kad neteisingai aptiktas atspindys, ekrano kraštas ar pašalinis objektas neturi būti priimamas kaip tinkamas registracijos pagrindas. Toks patikrinimas leidžia sumažinti klaidingų aptikimų tikimybę ir užtikrina, kad transformacija būtų skaičiuojama iš realių YOLO kaukių, o ne iš atsitiktinių kadro elementų.

4.3. ROI sudarymas normalizuotame vaizde

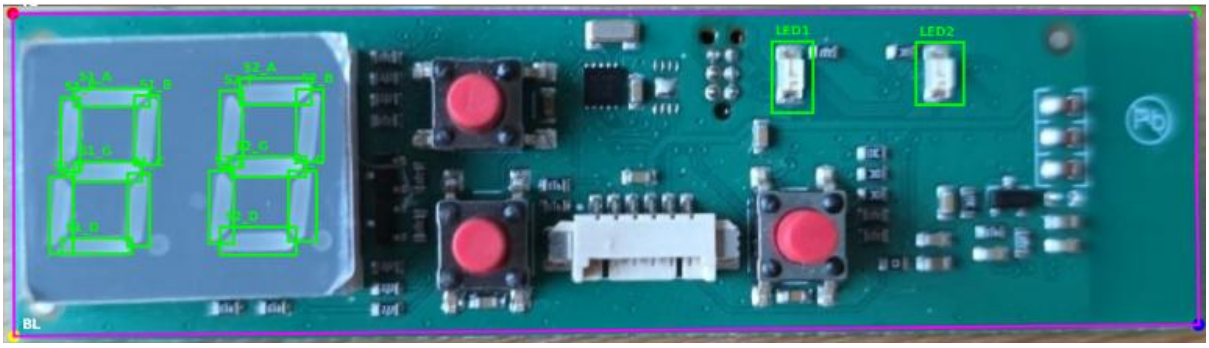
Po geometrijos suvienodinimo kiekvienas kadras perkeliamas į bendrą koordinačių sistemą, kurioje tiriamojo mazgo vaizdas užima apibrėžtą ir pakartojamą vietą. Tai leidžia ROI sritis apibrėžti ne gyvame 640 × 480 kadre, o normalizuotame vaizde, kuriame PCB orientacija ir mastelis yra suvienodinti. Toks sprendimas tiesiogiai mažina jautrumą kintančiai geometrijai ir yra analogiškas praktikoje sutinkamam reikalavimui tiksliai suderinti vaizdus prieš lyginimą ar klasifikaciją (pvz., „Golden-board“ inspekcijoje) [11].

ROI sričių sudarymas atliktas remiantis etaloniniu kadru, kuriame tiriamasis objektas buvo aiškiai matomas, o segmentinio ekrano ir papildomų LED padėtys buvo pažymėtos rankiniu būdu. Šiame etaloniniame kadre buvo apibrėžtos 17 pagrindinių sričių: 14 segmentų sričių, atitinkančių „A1“ – „G1“ ir „A2“ – „G2“ segmentus, viena bendra PCB sritis ir dvi papildomų LED sritys. Kiekviena ROI sritis aprašyta keturkampiu, kurio koordinatės išsaugotos konfigūraciniame faile. Taigi, ROI išdėstymas nėra iš naujo žymimas kiekvienam kadrui, o vieną kartą apibrėžiamas etaloninėje koordinačių sistemoje ir vėliau perskaičiuojamas pagal aptiktą ekrano bei PCB padėtį.

ROI konfigūraciniame faile kiekvienai sričiai saugomas pavadinimas, tipas ir keturių kampų koordinatės etaloninėje koordinačių sistemoje. Segmentų pavadinimai atitinka fizinį ekrano išdėstymą: „A1“–„G1“ kairiajam skaitmeniui ir „A2“–„G2“ dešiniajam skaitmeniui. LED sritys aprašomos atskirai, nes jų sprendimo taisyklės grindžiamos spalviniais požymiais. Perskaičiavus ROI į naują kadrą tikrinama, ar visos sritys patenka į vaizdo ribas ir ar jų padėtis neprieštarauja aptikto ekrano bei PCB geometrijai.

Segmentų ekranui ir dviem papildomiems LED, rankiniu būdu, etaloniniame vaizde suformuojamos atskiros ROI sritys (žr. 20 pav.):

- segmentų ROI, kurios dengia kiekvieną šviečiantį segmentą atskirai;
- LED ROI, kurios dengia du papildomus LED;
- fono ROI, kuri parenkama iš srities, kurioje neturi būti švytėjimo (naudojama ryškumo kompensacijai).



20 pav. Kadro dominančios sritys (ROI)

Segmentų ROI dydžiai parinkti taip, kad sritis apimtų visą fizinį segmento plotą, bet kiek įmanoma mažiau persidengtų su gretimais segmentais. Tai svarbu todėl, kad 7 segmentų ekrane „A“ – „G“ segmentai išsidėstę arti vienas kito, o per didelė ROI sritis gali įtraukti gretimo segmento švytėjimą arba atspindį. Dėl šios priežasties galutiniam būsenos sprendimui naudojama ne visa pažymėta segmento ROI, o vidinė jos dalis. Horizontaliems segmentams „A“, „D“ ir „G“ vertinimo sritis susiaurinama 75 proc. iš abiejų pusių pagal *X* kryptį iki vidurinės segmento dalies, o vertikaliems segmentams „B“, „C“, „E“ ir „F“ – analogiškai pagal *Y* kryptį. Taip sumažinama klaidingo įjungimo tikimybė, kai šviesa ar atspindys patenka į segmento kraštus. Vizualizacijoje gali būti rodomas visas segmento ROI, tačiau požymiai skaičiuojami iš vidinės stabiliausios srities.

ROI padėties paklaida vertinama ne kaip atskiras geometrinis matavimas milimetrais, o pagal tai, ar perskaičiuotos ROI sritys naujame kadre sutampa su realiomis segmentų ir LED vietomis. Tam naudojama vizualinė kontrolė ir klaidingų atpažinimų analizė. Jeigu ROI perskaičiavimas netikslus ir sritis patenka ne į segmentą ar LED, o į ekrano kraštą, foną arba atspindžio zoną, toks atvejis priskiriamas geometrijos arba ROI pozicionavimo klaidai. Jeigu ROI yra tinkamoje vietoje, tačiau būsena nustatyta klaidingai dėl šviesos atspindžio, spalvinio iškraipymo ar ryškio pokyčio, toks atvejis priskiriamas optinių trikdžių sukeltai būsenos nustatymo klaidai.

Toks ROI sudarymo būdas pasirinktas todėl, kad leidžia atskirti du etapus: vieną kartą apibrėžtą fizinių indikatorių išdėstymą etaloniniame kadre ir automatinį šio išdėstymo perkėlimą į naujus kadrus. Taip galima vertinti ne tik galutinį atpažinimo tikslumą, bet ir tai, ar klaida atsirado dėl netikslaus ROI perkėlimo, ar dėl pačioje ROI srityje esančių šviesumo bei spalvos požymių.

4.4. Požymių skaičiavimas ir būsenos sprendimas

Kadangi sistemoje naudojamas aplinkos apšvietimas ir geometrija kinta, vien tik absoliutus ROI ryškumas nėra pakankamai patikimas kriterijus. Praktikoje net ir su internetine kamera matoma, kad ekspozicija, triukšmas ir foninis apšvietimas gali lemti reikšmingus intensyvumo svyravimus, todėl sprendimo taisyklė turi turėti foninę kompensaciją. Tokį jautrumą apšvietimui ir kalibravimo svarbą pabrėžia ir praktiniai LED inspekcijos sprendimai, kur apšvietimas ir ekspozicija kalibruojami „juoda“ / „balta“ principu, siekiant išvengti klaidingų rezultatų [19].

Šiame projekte kiekvienai ROI sričiai apskaičiuojamas intensyvumo požymis, o sprendimas priimamas ROI ryškumą palyginus su fono ryškumu. Minimalus, bet pakankamai stabilus požymis yra ROI vidutinė (arba mediana) pilkumo reikšmė. Tada kiekvienam segmentui taikomas kriterijus:

- segmentas / LED laikomas įjungtu, jei $I_{ROI} - I_B > T$;
- segmentas / LED laikomas išjungtu, jei $I_{ROI} - I_B \leq T$;

čia I_{ROI} – tai ROI intensyvumas, I_B – tai fono ROI intensyvumas, o T – slenkstis.

Segmentas arba papildomas LED laikomas įjungtu, kai atitinkamos ROI srities ir fono ROI intensyvumo skirtumas viršija slenkstį *T*. Patikimumo kontrolei taikoma kadro kokybės patikra, registracijos kokybės patvirtinimas ir sprendimo aiškumo vertinimas. Aptikus neatitikimus, kadras žymimas klaidos kodu, o rezultatas laikomas negaliojančiu.

Kiekybiškai sprendimo taisyklė apibrėžiama skirtumu:

$$\Delta I = \bar{I}_{ROI} - \bar{I}_{fonas}; \tag{1}$$

čia $\bar{I}_{ROI}$ – segmento arba LED ROI vidutinis intensyvumas, o $\bar{I}_{fonas}$ – foninės ROI srities vidutinis intensyvumas. Jei $\Delta I > T$, elementas laikomas įjungtu, kitu atveju – išjungtu.

Bendras kadrų atpažinimo tikslumas apskaičiuotas pagal formulę:

$$A_k = \frac{N_{teis}}{N} \cdot 100\ \%; \tag{2}$$

o segmentų būsenų tikslumas:

$$A_s = \frac{N_{seg,teis}}{14N} \cdot 100\ \%; \tag{3}$$

čia A_k – bendras kadrų atpažinimo tikslumas, proc., N_{teis} – teisingai atpažintų kadrų skaičius, A_s – segmentų būsenų nustatymo tikslumas, proc., $N_{seg,teis}$ – teisingai nustatytų segmentų būsenų skaičius, N – analizuotų objektų skaičius, o 14 – viename kadre analizuojamų segmentų skaičius.

Slenkstis gali būti:
- bendras visam ekranui (paprastesnis, bet jautresnis netolygiam apšvietimui);
- atskiras segmentams ir LED (stabiliau, nes LED optika ir segmentų švytėjimas skiriasi).

Tokio tipo slenkstiniai ir taisyklėmis grįsti metodai yra klasikinė bazė pramoninei inspekcijai, tačiau literatūroje pabrėžiama, kad jie blogėja, kai atsiranda mažos detalės, netolygus apšvietimas arba sudėtingi fonai; dėl to dažnai pereinama prie pažangesnių metodų [12], [16].

Šiuo atveju siekiama ne idealios laboratorinės geometrijos, o metodo atsparumo, Tparinkimas turi būti atliekamas iš bandymų rinkinio, kuriame yra įvairios geometrijos ir skirtingi paros laikai (rytas/vakaras). Tai tiesiogiai dera su tuo, kaip literatūroje vertinamas sistemų patikimumas keičiantis sąlygoms (svarbu ne vienas gražus kadras, o stabilus veikimas). [16]

Po ROI sričių sudarymo kiekvienoje srityje apskaičiuojami požymiai, pagal kuriuos priimamas binarinis sprendimas: optinis elementas įjungtas arba išjungtas. Kadangi tiriamajame objekte naudojami dviejų tipų indikatoriai – balta spalva šviečiantys 7 segmentų ekrano segmentai ir spalviniai LED indikatoriai – jiems taikomos skirtingos požymių grupės. Segmentų būsenai svarbiausias yra šviesumas ir baltos šviesos požymis, o LED būsenai – atitinkamos spalvos dominavimas, spalvos sodrumas ir ryškumas.

Vaizdas analizuojamas RGB (raudona, žalia, mėlyna, angl. *red, green, blue*) ir HSV (atspalvis, sodrumas, ryškumas, angl. *hue, saturation, value*) spalvinėse erdvėse. RGB kanalai naudojami spalvų santykiams ir kanalų sklaidai įvertinti, o HSV erdvė naudojama ryškiui Vir sodrumui Sapskaičiuoti. Segmentų ROI srityse apskaičiuojamos vidutinės R, G, B, S ir V reikšmės. Kadangi segmentai šviečia artima baltai šviesa, jų įjungimo sprendime vertinamas ne vien absoliutus ryškis, bet ir RGB kanalų tarpusavio panašumas. Tam naudojamas RGB kanalų sklaidos požymis:

$$D_{RGB} = \frac{|R-G|+|R-B|+|G-B|}{3}; \tag{4}$$

čia R, G ir B – vidutinės raudono, žalio ir mėlyno spalvų kanalų reikšmės analizuojamoje ROI srityje, D_{RGB} – spalvų kanalų tarpusavio skirtumo įvertis. Kuo ši reikšmė mažesnė, tuo šviesa artimesnė baltai arba pilkai. Baltumo požymis apskaičiuojamas taip:

$$W = V - 0{,}5 \cdot D_{RGB}; \tag{5}$$

čia W – baltumo požymis, o V – šviesumo reikšmė. Toks požymis leidžia atskirti tikrai šviesų segmentą nuo spalvinio atspindžio arba triukšmo. Galutiniam segmento vertinimui naudojamas sujungtas šviesumo ir baltumo įvertis:

$$Q_{seg} = 0{,}8 \cdot W + 0{,}2 \cdot V; \tag{6}$$

čia Q_{seg} – galutinis segmento aktyvumo įvertis. Segmentas laikomas įjungtu, jei tenkinamos trys sąlygos:

$$Q_{seg} \geq T_W;$$

$$V \geq T_V;$$

$$D_{RGB} \leq T_D;$$

čia T_W – baltumo įverčio slenkstis, T_V – minimalus ryškio slenkstis, o T_D – didžiausia leidžiama RGB kanalų sklaida. Slenksčiai buvo parinkti empiriškai programos kūrimo ir konfigūravimo metu, naudojant 15 pavienių segmentų kadrų derinimo imtį. Ši imtis buvo skirta ne galutiniam algoritmo tikslumui vertinti, o praktiniam ROI pozicionavimo, požymių skaičiavimo ir slenkstinės sprendimo logikos suderinimui realiomis kameros, apšvietimo ir tiriamojo objekto padėties sąlygomis. Segmentams taikyta ta pati sprendimo logika, nes visi 14 segmentų priklauso tam pačiam ekranui ir šviečia artima balta spalva. Galutinis algoritmo tikslumas skaičiuotas ne pagal derinimo imtį, o pagal atskirą 840 kadrų funkcinio vertinimo imtį.

LED indikatoriams taikoma atskira sprendimo logika, nes jų būsena priklauso ne tik nuo ryškio, bet ir nuo spalvos grynumo. LED1 vertinamas kaip raudonas indikatorius, o LED2 – kaip mėlynas indikatorius. Raudonam LED apskaičiuojami šie požymiai:

$$P_R = \frac{R}{R+G+B}; \tag{7}$$

$$K_{RG} = \frac{R}{G}; \tag{8}$$

$$K_{RB} = \frac{R}{B}; \tag{9}$$

čia P_R parodo, kokią visos spalvinės energijos dalį sudaro raudonas kanalas, o K_{RG} ir K_{RB} parodo, kiek raudonas kanalas viršija žalią ir mėlyną kanalus. Raudonas LED laikomas įjungtu tik tada, kai jis yra pakankamai ryškus, pakankamai sodrus ir raudonas kanalas aiškiai dominuoja prieš kitus kanalus. Praktinėje algoritmo versijoje naudojamos tokios sąlygos:

$$R \geq 70;$$

$$V \geq 50;$$

$$S \geq 80;$$

$$P_R \geq 0{,}52;$$

$$K_{RG} \geq 1{,}8;$$

$$K_{RB} \geq 1{,}8.$$

Mėlynam LED analogiškai skaičiuojamas mėlynos spalvos grynumas ir mėlyno kanalo santykis su kitais kanalais:

$$P_B = \frac{B}{R+G+B}; \quad (10)$$

$$K_{BR} = \frac{B}{R}; \quad (11)$$

$$K_{BG} = \frac{B}{G}; \quad (12)$$

čia P_B parodo, kokią visos spalvinės energijos dalį sudaro mėlynas kanalas, o K_{BR} ir K_{BG} parodo, kiek mėlynas kanalas viršija raudoną ir žalią kanalus. Mėlynas LED laikomas įjungtu, kai:

$$B \geq 70;$$

$$V \geq 50;$$

$$S \geq 80;$$

$$P_B \geq 0{,}52;$$

$$K_{BR} \geq 1{,}8;$$

$$K_{BG} \geq 1{,}8.$$

Tokios sąlygos įvestos todėl, kad vien spalvos dominavimo požymio nepakanka. Pavyzdžiui, išjungtas LED gali atrodyti šiek tiek rausvas arba melsvas dėl atspindžio nuo PCB, ekrano ar aplinkos apšvietimo. Tokiu atveju vien $R - G$, $R - B$ arba $B - R$ skirtumas gali būti teigiamas, nors LED realiai nešviečia. Dėl to vertinamas ne tik spalvos perteklius, bet ir spalvos grynumas, sodrumas bei kanalo santykis su kitais kanalais. LED būsena pripažįstama įjungta tik tada, kai visi šie požymiai vienu metu atitinka nustatytas ribas.

Ribiniai atvejai apibrėžiami pagal tai, ar požymiai yra arti slenksčių. Jeigu segmento arba LED požymis tik nedaug viršija slenkstį, toks sprendimas laikomas jautriu apšvietimo ir triukšmo pokyčiams. Praktinėje analizėje ribiniais laikomi atvejai, kai pagrindinis požymis nuo slenksčio skiriasi mažiau kaip 5 proc. Tokie kadrai papildomai vertinami klaidų analizėje, nes nedidelis ekspozicijos, atspindžio arba ROI padėties pokytis gali pakeisti galutinę būseną.

Sprendimas laikomas nepatikimu, jei tenkinama bent viena iš šių sąlygų: YOLO modelis neaptinka PCB arba segmentinio ekrano srities; ROI perskaičiavimas akivaizdžiai nepataiko į fizinį segmentą arba LED; segmentų ar LED požymiai yra ribinėje zonoje; vienoje ROI srityje atsiranda ryškus atspindys, kuris neatitinka tikro optinio elemento padėties; vienu metu aktyviais nustatomi keli fiziškai nesuderinami segmentai, kai valdiklio režime turėjo šviesti tik vienas segmentas. Tokie atvejai nėra šalinami iš tyrimo rezultatų, bet žymimi kaip klaidingi arba nepatikimi ir naudojami nustatant klaidų priežastis.

4.5. Programinės įrangos struktūra, kadrų surinkimo seka ir vaizdų analizės algoritmas su patikimumo kontrole

Šiame projekte programinė dalis sudaryta iš dviejų tarpusavyje susietų grandžių – valdiklio (ESP32-S3) ir kompiuterio programos „Python“. Valdiklis generuoja ekrano / LED būsenas, o kompiuteris

pagal jas fiksuoja kadrus ir vykdo vaizdų analizę. Svarbus šios sistemos bruožas – rankiniu būdu inicijuojami kadrų surinkimo seansai: vartotojas pats padeda PCB į norimą padėtį, paspaudžia „START“, sistema surenka visų būsenų kadrus ir sustoja. Toks darbo režimas leidžia valdyti geometrijos kaitą (pasukimai, poslinkiai, skirtingos dienos apšvietimas), o vėliau analizuoti rezultatų atsparumą sąlygoms, kurios realiai ir sukelia didžiąją dalį klaidų.

ESP32-S3 pusėje numatytas paprastas nuotolinio valdymo protokolas per UART, o valdiklio būsenų generavimo programos kodas pateiktas 1 priede. Kompiuteris siunčia dvi pagrindines komandas:

- „SETVALn“ – nustato 7 segmentų ekrano reikšmę (0–99);
- „SETLEDm“ – nustato dviejų papildomų LED būsenų režimą ($m \in \{0,1,2,3\}$), kai praktiškai gaunamos keturios kombinacijos (0/0; 1/0; 1/1; 0/1).

Taigi, vienas surinkimo seansas sudaro visų testuojamų ekranų būsenų aibę, sujungtą su LED kombinacijomis. Būsenų aibė šiame projekte apibrėžiama taip:

- 100 skaitinių būsenų nuo „00“ iki „99“;
- 14 segmentinių būsenų (kai įjungiami atskiri segmentai ir jų kombinacijos pagal pasirinktą sąrašą);
- kiekviena iš jų kartojama su 4 LED kombinacijomis, todėl gaunama 456 būsenų seka.

4.5.1. Integruotas kadrų surinkimo ir analizės algoritmas

Toliau pateikiamas integruotas algoritmas aprašo visą ciklą – nuo rankinio sistemos paleidimo (mygtuko „START“ paspaudimo) iki rezultatų išsaugojimo. Algoritmo struktūra specialiai daroma tokia, kad būtų aišku, kur prasideda ir baigiasi vienas seansas, kaip užtikrinamas LED nusistovėjimas prieš fiksavimą, kaip sprendžiama kintanti geometrija, kaip fiksuojami klaidų atvejai, kad sistemai suklydus, klaida būtų matoma.

4.5.2. Algoritmas: kadrų surinkimas, registracija, ROI formavimas ir būsenų įvertinimas

1. Seanso inicijavimas (rankiniu būdu). Vartotojas paruošia PCB padėtį (pasuka, pastumia, pakeičia atstumą nežymiai) ir programoje paspaudžia „START“.
2. Seanso katalogo sukūrimas. Sukuriamas naujas aplankas su unikaliu pavadinimu („data_laikas_valXX_seanso indeksas“). Į jį bus rašomi visi kadrai ir metaduomenys.
3. Resursų paruošimas.
   3.1. atidaromas UART ryšys su ESP32-S3.
   3.2. inicializuojama kamera su nustatytais parametrais (raiška / kadrų dažnis ir kt.).
4. Būsenų sąrašo sudarymas. Programoje sugeneruojamas būsenų masyvas, sudarytas iš skaitinių ir segmentinių būsenų, o kiekvienai būsenai priskiriamos 4 LED kombinacijos (iš viso 456 kadrai).
5. Ciklas per būsenas (viename seanse). Kiekvienai būsenai *i* atliekami 5.1–5.9 žingsniai:
   5.1. per UART išsiunčiama komanda SETVAL (kai būsena skaitinė) arba atitinkama ekrano būsenos komanda (kai būsena segmentinė), ir SETLEDm.
   5.2. laukiama 200 ms, kad LED švytėjimas nusistovėtų.
   5.3. iš kameros paimamas vienas kadras.
   5.4. kadras išsaugomas seanso aplanke su vardu, kuriame aiškiai užkoduota būsena („yymmdd_valXX_ledXX_XXXX.png“).
   5.5. į „log“ failą įrašomi metaduomenys: būsena, LED režimas, laiko žyma, failo vardas.
   5.6. (analizė – realiuoju laiku arba ne) kadras perduodamas vaizdų analizės grandinei:
      5.6.1. vaizdas paruošiamas (spalvų konversija, triukšmo slopinimas).

5.6.2. iš YOLOv5 segmentavimo modelio rezultatų gaunamos PCB ir segmentinio ekrano sritys.
5.6.3. pagal aptiktą segmentinio ekrano ir PCB geometriją ROI sritys perskaičiuojamos etaloninės į einamojo kadro koordinates.
5.6.4. pagal registraciją suformuojamos ROI sritys ekrano segmentams ir dviem LED.
5.6.5. apskaičiuojami intensyvumai ROI srityse ir priimamas sprendimas (ON / OFF).
5.6.6. rezultatas lyginamas su etalonine būsena (pagal tą pačią būseną, kuri buvo išsiųsta į valdiklį).
5.6.7. apskaičiuojamas sprendimo patikimumas (pvz., skirtumas tarp ROI ir fono, ribinių atvejų skaičius).
5.6.8. jei nustatoma apdorojimo klaida, pavyzdžiui, neaptinkama PCB arba ekrano sritis, ROI išeina už kadro ribų arba gaunamas ribinis sprendimas, kadrui priskiriamas atitinkamas klaidos kodas. Jeigu kadro būsena gali būti apskaičiuota, bet nesutampa su etalonu, jis skaičiuojamas kaip klaidingas atpažinimas. Jeigu dėl aptikimo ar geometrijos klaidos būsena negali būti patikimai nustatyta, kadras žymimas kaip techninė apdorojimo klaida ir atskirai įtraukiamas į klaidų priežasčių analizę.
5.6.9. laukiama 400ms (pauzė tarp būsenų po nusistovėjimo), tada pereinama prie kitos būsenos.

6. Seanso pabaiga. Pasibaigus 456 būsenų ciklui, programa uždaro UART ir kameros resursus, sugeneruoja seanso suvestinę (kiek kadrų sėkmingi, kiek atmesti, kokių klaidų kodai dominavo) ir sustoja.
7. Pakartotinis seansas. Vartotojas, esant poreikiui, pakeičia PCB padėtį / geometriją ir vėl paspaudžia „START“ – ciklas kartojamas nuo 1 žingsnio.

Kadrų surinkimo, UART ryšio ir etaloninių būsenų registravimo programos kodas pateiktas 2 priede.

4.5.3. Kintančios geometrijos sprendimas: registracija pagal PCB orientyrus ir homografija

Šiame projekte atsisakoma fiksuoto ROI, nes PCB padėtis nėra pastovi – ji kinta kiekviename seanse. Todėl ROI formuojamas tik po registracijos, kai surandami PCB orientyrai. Praktinis ir stabilus pasirinkimas yra plokštės kampai: jų geometrija nekinta, jie mažai priklauso nuo ekrano švytėjimo ir leidžia atkurti plokštumos padėtį. Homografijos taikymas leidžia kompensuoti poslinkį, pasukimą ir perspektyvinį iškraipymą, o tai yra tiesioginis atsakas į situacijas, kai geometrija kintanti, o ne laboratorinė.

Kadangi vien PCB stačiakampis gali būti dviprasmiškas (šalia esantys du LED, apvertimo atvejai), segmentų ekranas su homografija suteikia aiškesnį registracijos pagrindą. Jei vis dėlto pasitaiko sugadintų atvejų (pvz., ieškant ekrano aptinkamas klaidingas elementas), registracijos patikimumas tikrinamas papildomomis sąlygomis:

- tikrinamas rasto segmentų ekrano ir PCB tarpusavio atstumų santykiai;
- tikrinama, ar transformacijos rezultatas logiškas (ROI sritys nepabėga už vaizdo ribų);
- esant abejonėms, kadras atmetamas ir pažymimas klaidos kodu.

4.5.4. Patikimumo kontrolė ir klaidų fiksavimas

Kad sistema būtų patikima realiomis sąlygomis, nepakanka vien atpažinti. Reikia turėti mechanizmą, kuris atskiria kada algoritmas suklydo nuo to, kada kadrą sugadino sąlygos. AGA dalyje akcentuojama, kad kameros nustatymai ir ekspozicijos valdymas tiesiogiai veikia patikimumą, o

praktiniuose bandymuose su internetinėmis kameromis galima pasiekti stabilių rezultatų tik aiškiai suvaldžius apšvietimo / ekspozicijos parametrus [16].

Šiame projekte patikimumo kontrolė įgyvendinama trimis lygiais:

- kadro kokybė (per tamsu / per šviesu / per mažas kontrastas);
- registracijos kokybė (ar tikrai teisingai „atsistojome“ į PCB koordinates);
- sprendimo aiškumas (ar segmentų / LED ROI intensyvumas pakankamai atsiskiria nuo fono).

Kad klaidų analizė nebūtų vertinama tik bendrai, kiekvienam nesėkmingam arba klaidingai įvertintam kadrui priskiriamas klaidos tipas. Šiame projekte klaidos skirstomos į dvi grupes: technines apdorojimo klaidas, kai kadro neįmanoma patikimai įvertinti dėl aptikimo ar geometrijos problemos, ir atpažinimo klaidas, kai kadras apdorotas, tačiau gauta būsena nesutampa su valdiklio sugeneruota etalonine būsena (žr. 8 lentelę).

8 lentelė. Klaidų kodų sistema ir jų įtraukimas į statistiką

Klaidos kodas	**Klaidos reikšmė**	**Priežastis**	**Įtraukimas į statistiką**
E0	Klaidos nėra	PCB, ekrano sritis ir ROI nustatyti korektiškai, rezultatas sutampa su etalonu	Skaičiuojamas kaip teisingas kadras
E1	Neaptikta PCB sritis	YOLO modelis neaptiko PCB arba aptikimas buvo nepakankamo patikimumo	Kadras žymimas kaip techninė aptikimo klaida; į bendrą tikslumo skaičiavimą gali būti įtraukiamas kaip neatpažintas kadras
E2	Neaptikta segmentinio ekrano sritis	YOLO modelis neaptiko ekrano arba aptikta sritis netinkama ROI formavimui	Kadras žymimas kaip techninė aptikimo klaida; rezultatas nelaikomas patikimu
E3	Netinkamas ROI perkėlimas	ROI sritys patenka už kadro ribų arba akivaizdžiai nesutampa su realia segmento / LED vieta	Kadras priskiriamas geometrijos arba ROI pozicionavimo klaidai
E4	Segmento būsenos klaida	ROI nustatyta tinkamai, tačiau bent vieno segmento „ON“ / „OFF“ būsena nesutampa su etalonu	Kadras skaičiuojamas kaip klaidingas atpažinimas; įtraukiamas į segmentų klaidų analizę
E5	LED būsenos klaida	LED ROI nustatyta tinkamai, tačiau LED 1 arba LED 2 būsena nesutampa su etalonu	Kadras skaičiuojamas kaip klaidingas atpažinimas; įtraukiamas į LED klaidų analizę
E6	Ribinis arba nepatikimas sprendimas	Požymiai yra arti slenksčio, yra stiprus atspindys arba viename kadre gaunama fiziškai abejotina būsena	Kadras papildomai pažymimas kaip nepatikimas; pagal galutinę būseną jis gali būti priskirtas teisingam arba klaidingam atpažinimui

Galutiniame tikslumo vertinime kadras laikomas teisingai atpažintu tik tada, kai visos jo vertinamos segmentų ir LED būsenos sutampa su etalonine valdiklio būsena. Jeigu bent vienas segmentas arba LED nustatomas klaidingai, kadras įtraukiamas į klaidingų atpažinimų skaičių. Techninės aptikimo ar ROI pozicionavimo klaidos nėra nutylimos: jos žymimos atskirais kodais ir naudojamos klaidų priežasčių analizei. Tokia tvarka leidžia atskirti, ar neteisingas rezultatas atsirado dėl objekto lokalizavimo, geometrijos suvienodinimo, ROI perkėlimo ar dėl pačios „ON“ / „OFF“ sprendimo taisyklės.

4.6. Apibendrinimas

Šiame skyriuje aprašytas vaizdų apdorojimo algoritmas, kuriuo iš vieno kameros kadro nustatomos 2 skaitmenų 7 segmentų ekrano ir dviejų papildomų LED būsenos. Algoritmo įėjimas yra kameros kadras, YOLOv5 segmentavimo modelis ir etaloninis ROI išdėstymo failas. YOLOv5 modelis naudojamas dviem sritims aptikti: PCB plokštei ir segmentinio ekrano sričiai. Pagal šias aptiktas sritis apskaičiuojama plokštės ir ekrano padėtis kadre, o etaloninės ROI sritys perkeliamos į realų kadrą.

ROI struktūra sudaryta iš 14 segmentų sričių, atitinkančių „A1“ – „G1“ ir „A2“ – „G2“ segmentus, bei dviejų LED sričių. Segmentų būsenoms nustatyti vertinami šviesumo ir baltumo požymiai: V kanalo reikšmė, RGB kanalų sklaida ir iš jų apskaičiuotas baltumo įvertis. Kad gretimų segmentų švytėjimas mažiau veiktų rezultatą, požymiai skaičiuojami ne iš viso segmento ROI, o iš vidinės jo dalies. LED 1 ir LED 2 vertinami atskirai: LED 1 tikrinamas pagal raudonos spalvos grynumą, sodrumą ir ryškumą, o LED 2 – pagal mėlynos spalvos grynumą, sodrumą ir ryškumą. Taip LED atveju sumažinama klaida, kai išjungtas LED atrodo rausvas arba melsvas dėl atspindžio.

Programos rezultatas yra 16 binarinių reikšmių rinkinys: 14 segmentų būsenų ir 2 LED būsenos. Kiekviena reikšmė įgyja 0 arba 1, t. y. optinis elementas laikomas išjungtu arba įjungtu. Šis rezultatas lyginamas su ESP32-S3 valdiklio sugeneruota etalonine būsena. Kadangi valdiklis iš anksto nustato, kuris segmentas arba LED turi šviesti, kiekvienam kadrui galima automatiškai priskirti teisingą atsakymą be rankinio žymėjimo.

Skyriuje taip pat apibrėžta, kada kadras laikomas netinkamu arba klaidingu. Jei YOLOv5 neaptinka PCB arba ekrano srities, klaida priskiriama aptikimo etapui. Jei ROI po perkėlimo nepatenka į realią segmento ar LED vietą, klaida priskiriama geometrijos arba ROI pozicionavimo etapui. Jei ROI yra tinkamoje vietoje, bet būsena nustatoma klaidingai, klaida siejama su požymių skaičiavimu, slenksčiais, atspindžiais arba apšvietimu. Toks klaidų atskyrimas leidžia 5 skyriuje vertinti ne tik bendrą tikslumą, bet ir nustatyti, dėl ko atsiranda klaidingi atpažinimai: dėl objekto padėties, ROI perkėlimo ar pačios „ON“ / „OFF“ sprendimo taisyklės.

5. Eksperimentinis tyrimas ir rezultatų aptarimas

5.1. Eksperimentų metodai

Toliau vykdomi eksperimentiniai tyrimai, skirti sukurti vaizdo apdorojimo metodą, leidžiantį automatiškai aptikti elektroninės plokštės (PCB) elementus ir nustatyti septynių segmentų indikatoriaus būseną naudojant kameros vaizdus. Darbai buvo vykdomi keliais etapais, pradedant duomenų paruošimu ir baigiant prototipinio analizės algoritmo kūrimu bei testavimu.

Eksperimentinis tyrimas buvo vykdomas keliais etapais. Pirmajame etape buvo surinktas pirminis 6 840 kadrų rinkinys, apimantis skaitines dviejų segmentų ekrano reikšmes ir pavienių segmentų būsenas su skirtingomis LED indikatorių kombinacijomis. Šis rinkinys buvo sudarytas iš 15 bandymų sesijų, kiekvienoje fiksuojant 456 būsenų kombinacijas. Pirminis rinkinys buvo naudojamas sistemos veikimui patikrinti, vaizdo fiksavimo sąlygoms įvertinti ir algoritminės grandinės kūrimui.

Vėlesniame tyrimo etape skaitinių reikšmių kaip simbolių atpažinimo atsisakyta, nes galutinis darbo vertinimas orientuotas į pavienių optinių elementų „ON“ / „OFF“ būsenų nustatymą. Programos kūrimo ir konfigūravimo metu papildomai naudota 15 pavienių segmentų kadrų derinimo imtis, skirta ROI pozicionavimui, slenkstinėms reikšmėms ir būsenų sprendimo logikai patikrinti. Ši imtis nebuvo laikoma galutiniu testavimo rinkiniu. Baigus programos konfigūravimą, galutinis sukurto algoritmo tikslumas įvertintas pagal 840 kadrų imtį.

Galutiniame vertinime buvo keičiama tiriamojo objekto padėtis kameros kadre, segmento būsena ir LED indikatorių kombinacija. Pastoviais laikyti tiriamasis objektas, kamera, valdiklio sudaroma etaloninė būsena ir tas pats vaizdų apdorojimo algoritmas. Vertinti rodikliai: bendras kadrų lygio tikslumas, segmentų „ON“ / „OFF“ būsenų tikslumas, LED indikatorių būsenų tikslumas ir klaidingų aktyvių segmentų pasiskirstymas.

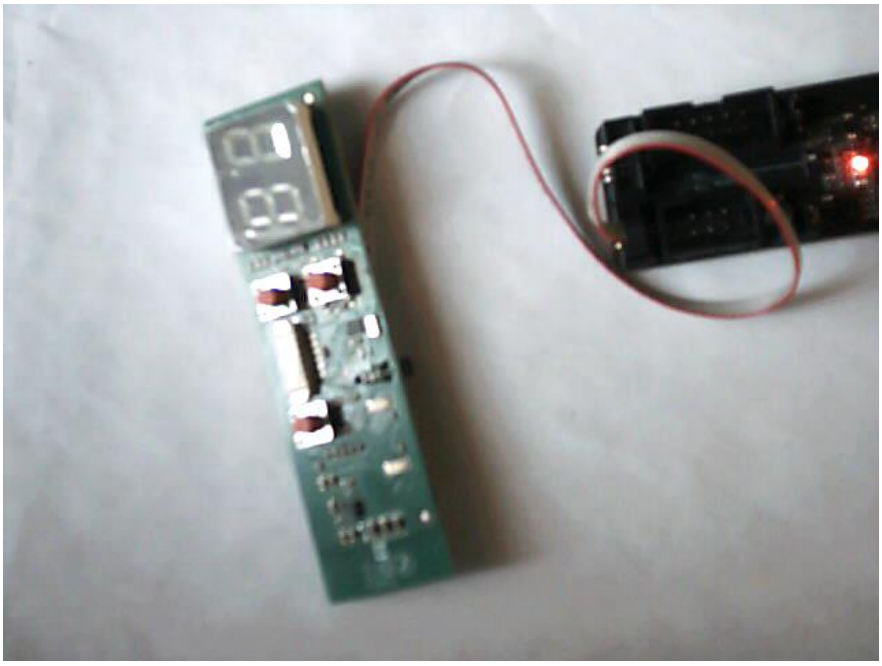
5.2. Duomenų rinkinys

Pirmiausia buvo parengtas YOLOv5 lokalizavimo modelio mokymo duomenų rinkinys. Tam iš pirminio 6 840 kadrų rinkinio atsitiktine tvarka atrinkti ir sužymėti 75 vaizdai. Atrankoje naudoti skirtingų bandymų sesijų kadrai, kuriuose tiriamasis objektas matomas esant skirtingai padėčiai, pasukimui ir apšvietimo sąlygoms. Į šią imtį įtraukti tiek vaizdai su skaitinėmis segmentinio ekrano reikšmėmis, tiek pavienių segmentų bandymų kadrai (žr. 21 pav.).

Vaizdai rankiniu būdu sužymėti naudojant „Label Studio“ anotavimo įrankį. Anotavimo metu pažymėtos dvi klasės: PCB, apibrėžianti visą tiriamosios plokštės regioną, ir SEG, apibrėžianti segmentinio ekrano sritį. Šios anotacijos skirtos tik objekto lokalizavimo modeliui mokyti, o ne galutinėms segmentų ar LED indikatorių „ON“ / „OFF“ būsenoms klasifikuoti. Anotacijų kokybė patikrinta vizualiai peržiūrint sužymėtus vaizdus ir įvertinant, ar PCB ir SEG sritys pažymėtos pagal realias tiriamojo objekto ribas.

Iš 75 sužymėtų vaizdų 60 naudota YOLOv5 modelio mokymui, o 15 – validacijai. Anotacijos eksportuotos į YOLO formatą ir naudotos PCB bei SEG klasių aptikimo modeliui mokyti. Atskiro YOLOv5 testavimo rinkinio nesudaryta, nes šiame darbe neuroninis tinklas naudotas ne galutinei segmentų ar LED būsenų klasifikacijai, o tik PCB ir segmentinio ekrano sričių lokalizavimui. Modelio lokalizavimo kokybė vertinta pagal validacijos imtį, o galutinis optinių indikatorių būsenų

atpažinimo tikslumas vertintas atskirai, naudojant visą YOLOv5, ROI ir slenkstinės logikos algoritminę grandinę.



21 pav. Duomenų rinkinio kadras

Programos kūrimo metu papildomai naudota 15 pavienių segmentų kadrų derinimo imtis. Ji buvo skirta praktiniam ROI išdėstymo, slenkstinių reikšmių ir būsenų sprendimo logikos patikrinimui programos kūrimo metu. Ši imtis nelaikyta galutiniu testavimo rinkiniu. Galutinis algoritmo veikimo vertinimas atliktas tik pabaigus programos konfigūravimą, naudojant 840 kadrų imtį.

Galutiniam vertinimui naudota 840 kadrų imtis buvo sudaryta iš pavienių segmentų bandymų kadrų. Joje vertintos 14 segmentų „ON“ / „OFF“ būsenos, keturios LED indikatorių kombinacijos ir 15 pakartojimų, iš viso 14 × 4 × 15 = 840 kadrų. Šioje imtyje nebuvo vertinamas skaitinės reikšmės kaip simbolio atpažinimas, nes galutinis tyrimas orientuotas į atskirų optinių elementų funkcinių būsenų nustatymą iš kameros vaizdo.

5.3. DNT apmokymas

Antrajame etape buvo apmokytas giliojo mokymosi modelis, naudojant YOLOv5 segmentacijos architektūrą. Modelio mokymui pasirinktas iš anksto apmokytas „yolov5s-seg“ modelis, kuris papildomai treniruotas naudojant 5.2 poskyryje aprašytą PCB ir SEG klasių duomenų rinkinį. Šiame darbe YOLOv5 modelis naudotas ne galutinei segmentų ar LED indikatorių būsenai klasifikuoti, o tiriamojo objekto geometrijai nustatyti – PCB ir segmentinio ekrano sritims lokalizuoti.

Mokymo procesas vykdytas 100 epochų, naudojant 640 × 640 pikselių įvesties vaizdus ir 8 vaizdų paketą. Naudoti iš anksto apmokyto modelio pradiniai svoriai, standartinės YOLOv5 duomenų augmentacijos, validacijos metrikų skaičiavimas po epochų ir geriausio svorių failo parinkimas pagal validacijos rezultatus. Modelio mokymas atliktas kompiuteriu „MSI Thin 15 B13VE“, turinčiu 13-os kartos „Intel Core i7-13620H“ procesorių, 16 GB RAM ir „NVIDIA GeForce RTX 4050 Laptop GPU“ 6 GB vaizdo plokštę (žr. 4 lentelę) [41].

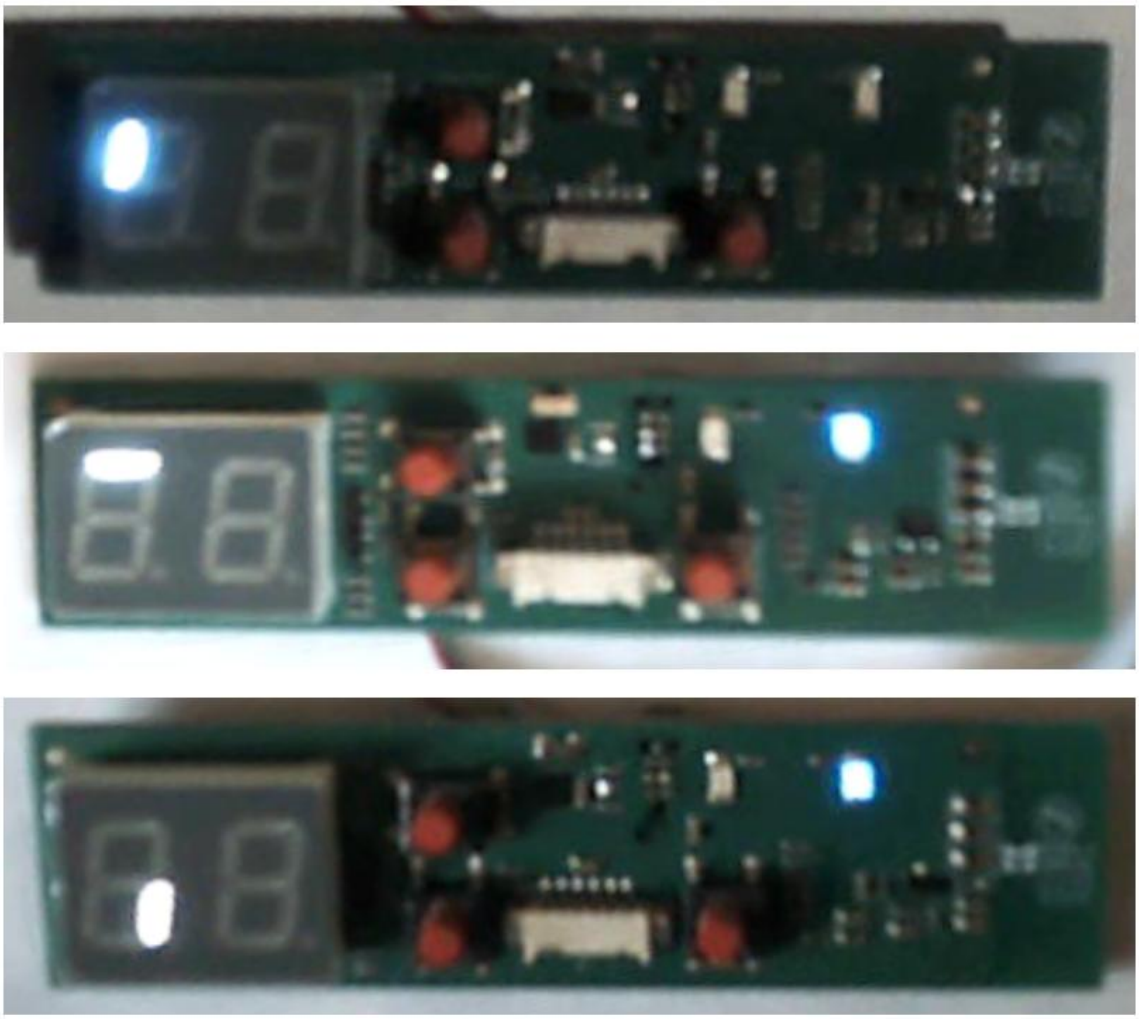
Gauti mokymo ir validacijos rezultatai pateikti 22 paveiksle. Paskutinėje mokymo epochoje objektų ribojančių sričių tikslumas validacijos imtyje siekė 0,9901, atkūrimas – 1,0000, mAP@0,5 – 0,9950, o mAP@0,5:0,95 – 0,8918. Segmentacijos kaukių tikslumas siekė 0,9901, atkūrimas – 1,0000, mAP@0,5 – 0,9950, o mAP@0,5:0,95 – 0,9179. Šie rodikliai rodo, kad PCB ir SEG sritys validacijos imtyje buvo lokalizuojamos patikimai.

22 pav. DNT YOLOv5 apmokymo rezultatai

Šios metrikos apibūdina tik YOLOv5 lokalizavimo modelio veikimą. Jos nereiškia galutinio segmentų ar LED indikatorių „ON“ / „OFF“ būsenų atpažinimo tikslumo. Galutinis būsenų atpažinimo tikslumas vertintas atskirai, taikant visą YOLOv5, ROI ir slenkstinės logikos grandinę 840 kadrų funkcinio vertinimo imčiai.

5.4. Kadro normalizavimas

Trečiajame etape buvo kuriamas vaizdo apdorojimo kodas, kuris naudoja apmokytą modelį realių vaizdų analizei. Buvo parašyti „Python“ programinės įrangos programiniai kodai, kurie įkelia modelį, atlieka PCB ir SEG objektų aptikimą bei iš aptiktos segmentacijos kaukės nustato ekrano regiono padėtį kadre. Remiantis aptiktos kaukės koordinatėmis, buvo realizuotas PCB normalizavimo algoritmas, leidžiantis iškirpti ir suvienodinti plokštės orientaciją taip, kad ji būtų pateikiama horizontaliai vienodoje padėtyje (žr. 23 pav.).



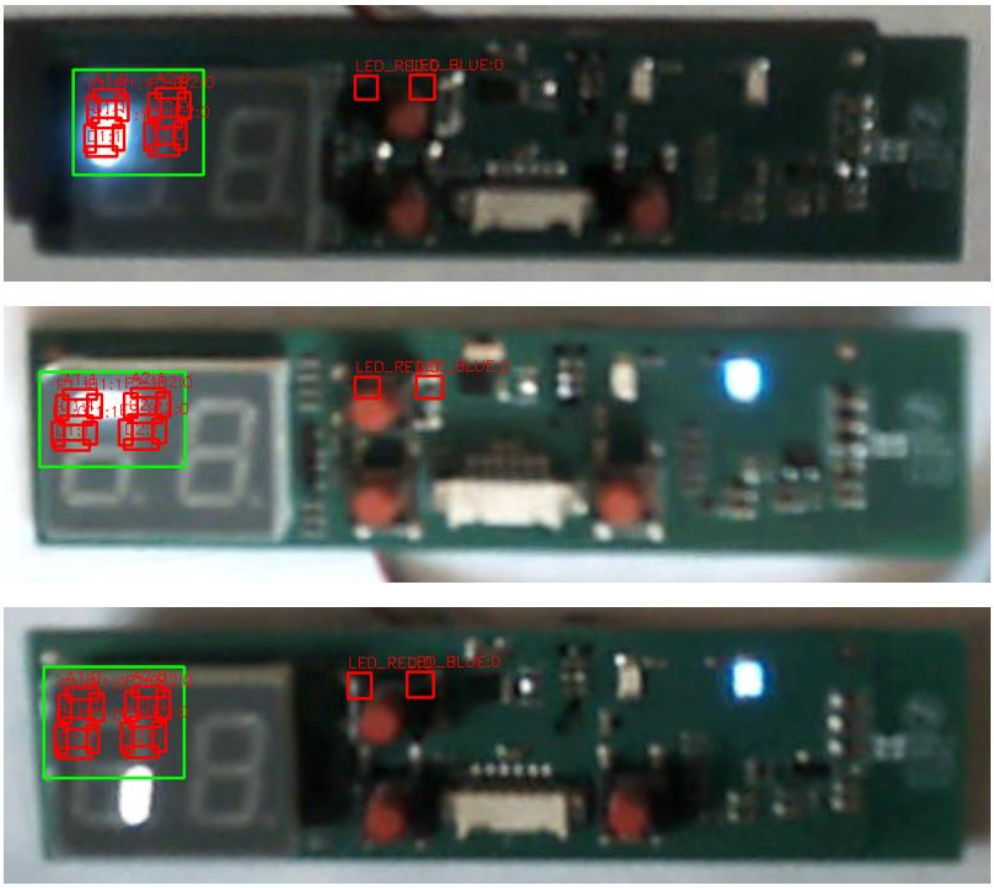
23 pav. Normalizuoti kadrai

Taip gauti normalizuoti PCB vaizdai, kuriuose indikatoriaus ekranas visuose kadruose yra maždaug toje pačioje vietoje ir orientacijoje.

5.5. ROI pritaikymas

Ketvirtajame etape buvo pradėtas ROI (dominančių sričių, angl. *region of interest*) išdėstymo kūrimas, siekiant automatiškai nustatyti konkrečių indikatoriaus elementų būseną. Remiantis etaloniniu kadru buvo pažymėtos ekrano, segmentų ir LED indikatorių sritys. Šios sritys buvo aprašytos konfigūraciniame faile kaip ROI koordinatės. Toliau buvo kuriamas algoritmas, kuris aptikęs SEG regioną naujame kadre pritaiko ROI išdėstymą tam regionui, proporcingai perskaičiuodamas koordinates pagal aptikto ekrano dydį. Taip siekiama užtikrinti, kad segmentų analizės sritys visada sutaptų su realia indikatoriaus padėtimi vaizde.

Be ROI pozicionavimo buvo pradėta ir segmentų būsenos nustatymo logikos realizacija. Kiekviename ROI regione apskaičiuojama pikselių intensyvumo vidutinė reikšmė, kuri naudojama sprendžiant, ar segmentas arba LED yra įjungtas („1“) ar išjungtas („0“). Taip suformuojamas dvejetainis indikatoriaus būsenos aprašas, kuris vėliau gali būti naudojamas skaičiaus ar indikatoriaus būsenos interpretavimui (žr. 24 pav.).



24 pav. ROI pozicionavimas kadruose

Eksperimentų metu buvo palyginti keli ROI transformavimo būdai: ROI taikymas pagal fiksuotas koordinates, ROI perskaičiavimas tik pagal SEG srities mastelį, orientacijos nustatymas pagal minimalų apibrėžiantį stačiakampį ir ROI perkėlimas pagal PCB bei ekrano kaukių tarpusavio geometriją. Fiksuotos koordinatės atmestos dėl jautrumo PCB pasislinkimui, vien SEG mastelio metodas – dėl netikslaus LED sričių perkėlimo, o minimalus stačiakampis – dėl nestabilios orientacijos, kai keičiasi atspindžiai arba ekrano kraštų matomumas. Galutiniam algoritmui pasirinktas PCB ir ekrano kaukių geometrija pagrįstas ROI perkėlimas, nes jis geriausiai išlaikė segmentų ir LED sričių sutapimą su realiomis jų padėtimis skirtinguose kadruose.

ROI pozicionavimo patikimumas šiame darbe vertintas ne kaip atskiras rankiniu būdu pažymėtų segmentų geometrijos palyginimas, o kaip visos lokalizavimo ir būsenų nustatymo grandinės dalis. Kadangi segmentų ROI buvo formuojamos pagal YOLOv5 aptiktą SEG sritį ir jos padėtį PCB atžvilgiu, pagrindiniu ROI inkaravimo patikimumo rodikliu laikyta SEG lokalizavimo kokybė validacijos imtyje. 5.3 poskyryje pateikti YOLOv5 rezultatai parodė aukštą PCB ir SEG sričių lokalizavimo tikslumą, todėl galutinio vertinimo klaidos pirmiausia siejamos ne su sisteminiu ROI poslinkiu, o su lokaliais optiniais trikdžiais: atspindžiais, netolygiu šviesumu ir kameros ekspozicijos prisitaikymu.

5.6. Tyrimo rezultatai

Atlikus sukurto vaizdo apdorojimo algoritmo eksperimentinį tyrimą, rezultatai vertinti keliais lygiais: kadrų lygiu, aktyvaus segmento nustatymo lygiu, visų segmentų „ON“ / „OFF“ būsenų lygiu ir papildomų LED būsenų lygiu (žr. 9 lentelę). Toks skirstymas pasirinktas todėl, kad vienas klaidingai

atpažintas kadras nebūtinai reiškia, jog klaidingai nustatytos visos optinių elementų būsenos. Pavyzdžiui, kadras gali būti laikomas klaidingu, jei neteisingai nustatytas vienas segmentas, nors visi kiti segmentai ir LED tame pačiame kadre atpažinti teisingai.

Galutinis būsenų atpažinimo tikslumas šiame poskyryje vertintas pagal 5.1 ir 5.2 poskyriuose aprašytą 840 kadrų funkcinio vertinimo imtį. Ši imtis sudaryta iš pavienių segmentų bandymų kadrų, todėl rezultatuose vertinamas ne skaitinės reikšmės kaip simbolio atpažinimas, o atskirų segmentų ir LED indikatorių „ON“ / „OFF“ būsenų nustatymas.

Bendras kadrų atpažinimo tikslumas apskaičiuotas pagal formulę:

$$A_k = \frac{N_k - N_{kl}}{N_k} \cdot 100\ \%; \tag{13}$$

čia A_k – kadrų atpažinimo tikslumas, N_k – bendras testuotų kadrų skaičius, N_{kl} – klaidingai atpažintų kadrų skaičius. Galutiniam vertinimui naudota 840 kadrų imtis, sudaryta iš atskirų segmentų ir LED būsenų bandymų. Iš šios imties 7 kadrai buvo atpažinti klaidingai, todėl:

$$A_k = \frac{840-7}{840} \cdot 100\% = 99{,}17\ \%; \tag{14}$$

Šis 99,17 proc. rodiklis reiškia kadrų lygio tikslumą: kadras laikomas teisingai atpažintu tik tada, kai algoritmu nustatyta būsena sutampa su valdiklio sugeneruota etalonine būsena. Jeigu bent vienas vertinamas optinis elementas nustatomas klaidingai, kadras priskiriamas klaidingai atpažintų kadrų grupei.

Segmentų būsenų tikslumas skaičiuotas ne kadrų, o atskirų segmentų „ON“ / „OFF“ būsenų lygiu. Kadangi viename kadre vertinama 14 segmentų, bendras segmentų būsenų skaičius apskaičiuojamas taip:

$$N_s = N_k \cdot 14 = 840 \cdot 14 = 11\,760; \tag{15}$$

Segmentų būsenų tikslumas apskaičiuotas pagal formulę:

$$A_s = \frac{N_s - N_{s,kl}}{N_s} \cdot 100\ \%; \tag{16}$$

čia A_s – segmentų būsenų tikslumas, N_s – visų vertintų segmentų būsenų skaičius, $N_{s,kl}$ – klaidingai nustatytų segmentų būsenų skaičius. Gauta 99,89 proc. reikšmė rodo, kad vertinant ne visą kadrą, o kiekvieno segmento „ON“ / „OFF“ būseną atskirai, klaidų dalis buvo labai maža.

Papildomų LED būsenos vertintos atskirai, nes LED1 ir LED2 turi po vieną binarinę būseną kiekviename kadre. LED tikslumas apskaičiuotas pagal tą patį principą:

$$A_{LED} = \frac{N_k - N_{LED,kl}}{N_k} \cdot 100\ \%; \tag{17}$$

čia A_{LED} – LED būsenų tikslumas, $N_{LED,kl}$ – klaidingai nustatytų LED būsenų skaičius.

Tyrimo metu LED1 ir LED2 būsenos visuose 840 kadruose sutapo su etaloninėmis valdiklio būsenomis, todėl abiejų LED būsenų nustatymo tikslumas siekė 100 proc.

9 lentelė. Tiriamo metodo segmentų ir indikatorių atpažinimo rezultatų suvestinė

Rodiklis	**Reikšmė**
Testuotų kadrų skaičius	840
Klaidingai atpažintų kadrų skaičius	7
Kadrų lygio atpažinimo tikslumas	99,17 proc.
Aktyvaus segmento nustatymo tikslumas	99,17 proc.
Segmentų „ON“ / „OFF“ būsenų tikslumas	99,89 proc.
LED1 būsenos nustatymo tikslumas	100,00 proc.
LED2 būsenos nustatymo tikslumas	100,00 proc.

9 lentelėje pateikti rezultatai turi būti interpretuojami kaip galutinės, susiaurintos tyrimo apimties rezultatai. Jie apibūdina ne skaitinių reikšmių atpažinimą, o atskirų optinių elementų funkcinių būsenų nustatymą. Toks vertinimas tiesiogiai susijęs su darbo tikslu, nes sistemoje svarbiausia nustatyti, kurie konkretūs segmentai ir LED indikatoriai yra įjungti arba išjungti.

Toks pats tikslumas gautas ir vertinant aktyvių segmentų nustatymą, nes visi klaidingi kadrų atpažinimo atvejai buvo susiję su aktyvaus segmento supainiojimu. Vertinant visų 14 segmentų būsenas dvejetainiu principu, gautas 99,89 proc. tikslumas. Abiejų papildomų indikatorių – LED1 ir LED2 – būsenos visais atvejais nustatytos teisingai, todėl jų atpažinimo tikslumas siekė 100,00 proc.

Svarbiausias šio darbo tikslui yra ne vien kadrų lygio rodiklis, o segmentų ir LED būsenų sutapimas su etalonu. Iš 840 vertintų kadrų 833 kadruose visa nustatyta būsena sutapo su ESP32-S3 valdiklio sugeneruota etalonine būsena. Likusiuose 7 kadruose klaidos buvo susijusios tik su aktyvaus segmento nustatymu. Papildomų LED indikatorių būsenos visuose kadruose nustatytos teisingai, todėl praktinį algoritmo ribojimą sudarė ne LED atpažinimas, o pavieniai aktyvaus segmento supainiojimo atvejai.

Klaidų kodų pasiskirstymas galutinėje vertinimo imtyje buvo toks: E0 – 833 kadrai, E1 – 0, E2 – 0, E3 – 0, E4 – 7, E5 – 0, E6 – 0. Tai rodo, kad galutinėje imtyje nepasitaikė techninių PCB ar segmentinio ekrano neaptikimo klaidų, o visi klaidingi atvejai buvo priskirti segmento būsenos nustatymo klaidai.

Detalesnei rezultatų analizei įvertinta, kuriuose segmentuose pasitaikė klaidingų atpažinimo atvejų. Klaidingų priskyrimų pasiskirstymas pagal segmentus pateiktas 10 lentelėje.

10 lentelė. Klaidingų aktyvių segmentų atpažinimo atvejų pasiskirstymas

Tikrasis aktyvus segmentas	**Klaidingai priskirtas segmentas**	**Klaidų skaičius**
„A2“	„B2“	2
„G2“	„C2“	4
„B2“	Neaptikta	1

10 lentelės duomenys rodo, kad klaidos nebuvo tolygiai pasiskirsčiusios tarp visų 14 segmentų. Jos koncentravosi dešiniajame skaitmenyje: 4 kartus tikrasis „G2“ segmentas buvo priskirtas „C2“ segmentui, 2 kartus „A2“ segmentas priskirtas „B2“ segmentui, o 1 kartą „B2“ segmentas nebuvo

aptiktas. Toks pasiskirstymas leidžia klaidas sieti ne su bendru algoritmo nestabilumu, o su konkrečiomis optinėmis sąlygomis dešiniojo skaitmens ROI srityse.

„G2“ ir „C2“ segmentų supainiojimas siejamas su atspindžiais ekrano srityje ir šviesumo pasiskirstymo iškraipymu. „G2“ segmentas yra vidurinėje ekrano dalyje, o „C2“ – dešiniojo skaitmens apatinėje dešinėje dalyje. Kai ekrano paviršiuje atsiranda atspindys arba padidėja lokalaus ryškumo zona, „C2“ ROI srityje gali susidaryti pakankamas šviesumo požymis, nors realiai aktyvus yra „G2“ segmentas. Tokiu atveju algoritmo sprendimą lemia ne neteisinga segmentų struktūra, o tai, kad atspindys „C2“ srityje tampa panašus į aktyvaus segmento požymį. Ši klaida ypač tikėtina tada, kai kameros ekspozicija automatiškai prisitaiko prie bendro kadro ryškumo, o segmentų kontrastas ekrane tampa nevienodas.

„A2“ ir „B2“ segmentų supainiojimas gali būti aiškinamas kita priežastimi. Šie segmentai yra arti vienas kito viršutinėje dešiniojo skaitmens dalyje, todėl net nedidelė ROI padėties paklaida arba perspektyvos pokytis gali lemti, kad vertinimo sritis iš dalies priartėja prie gretimo segmento arba jo švytėjimo zonos. Jei tuo pačiu metu ekrane yra atspindys, šviesos perėjimas tarp „A2“ ir „B2“ sričių tampa mažiau aiškus, todėl vieno segmento aktyvumas gali būti priskirtas kitam. Vienas „B2“ neaptikimo atvejis rodo priešingą situaciją – aktyvaus segmento požymis ROI srityje buvo per silpnas arba iškraipytas, todėl neviršijo būsenos sprendimui taikyto slenksčio.

Vizualinė klaidingų kadrų analizė rodo, kad pagrindinė šių klaidų priežastis buvo ne pats segmentų išdėstymas ar bendras ROI sudarymo principas, o optinės vaizdo fiksavimo sąlygos. Atspindžiai, lokalaus apšvietimo nevienodumas ir kameros ekspozicijos prisitaikymas pakeitė segmentų ryškumo pasiskirstymą ROI srityse. ROI padėties paklaida šiuose atvejuose galėjo sustiprinti klaidą, tačiau pagal klaidų pobūdį ji nebuvo pagrindinis veiksnys. Todėl tolesniame algoritmo tobulinime pirmiausia reikėtų mažinti atspindžius ir stabilizuoti apšvietimą, o tik po to papildomai tikslinti ROI ribas arba segmentų slenksčius.

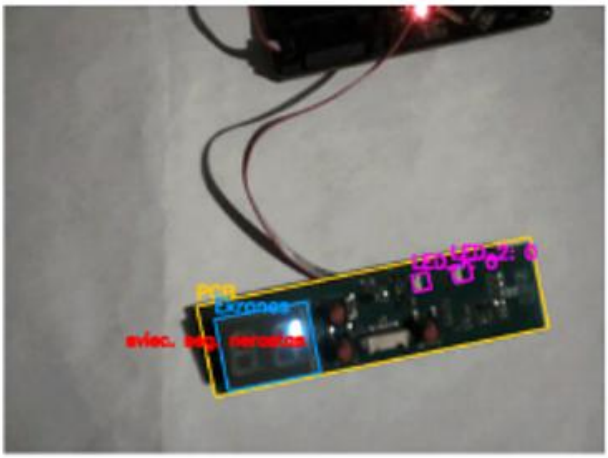
Siekiant detaliau įvertinti neteisingo atpažinimo priežastis, atlikta klaidingai klasifikuotų kadrų vizualinė analizė. Būdingi klaidingo aktyvių segmentų atpažinimo pavyzdžiai pateikti 25 paveiksle.

Klaidingo atpažinimo atvejai buvo susiję su atspindžiais ekrano srityje. Dėl jų pakito aktyvių segmentų ryškumo pasiskirstymas, todėl kai kuriais atvejais segmentas „G2“ buvo supainiotas su „C2“, „A2“ – su „B2“, o vienu atveju „B2“ segmentas nebuvo aptiktas.

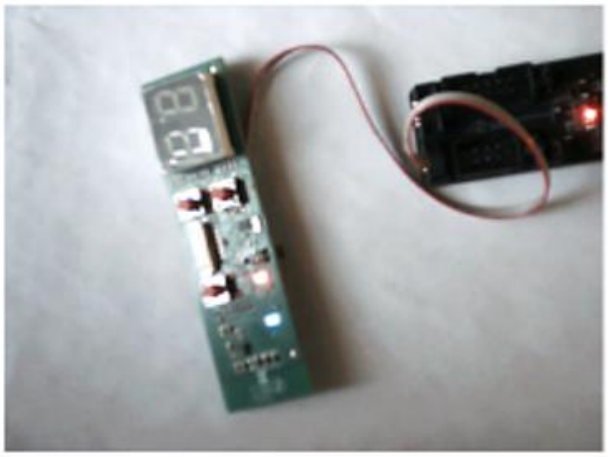
Tai rodo, kad pagrindinis metodą ribojantis veiksnys buvo ne bendra ROI lokalizacijos schema ar segmentų išdėstymo aprašymas, bet optiniai trikdžiai vaizdo fiksavimo metu. Vis dėlto konkrečių „A2“–„B2“ ir „G2“–„C2“ klaidų analizė parodė, kad atspindžiai pavojingiausi tada, kai jie sutampa su gretimo segmento ROI arba sumažina aktyvaus segmento kontrastą. Dėl to praktiniame taikyme tikslinga ne tik gerinti apšvietimą, bet ir papildomai tikrinti segmentų tarpusavio logiką, kai vieno aktyvaus segmento režime aptinkami keli panašaus patikimumo kandidatai.

Gautas 99,17 proc. kadrų lygio tikslumas rodo, kad sukurtas algoritmas pagal tikslumo reikšmę patenka į literatūroje aprašomų vaizdo analize pagrįstų optinių elementų atpažinimo metodų rezultatų intervalą. Pavyzdžiui, realaus laiko 7 segmentų LED ekrano atpažinimo sistemoje, kurioje taikytas CNN pagrįstas interpretavimo metodas, nurodytas 91,1 proc. atpažinimo tikslumas. Tuo tarpu transporto LED modulių atpažinimo tyrimuose, kuriuose naudotas modifikuotas YOLOv5 pagrindu veikiantis metodas, nurodomas 98,6–99,2 proc. tikslumo intervalas.

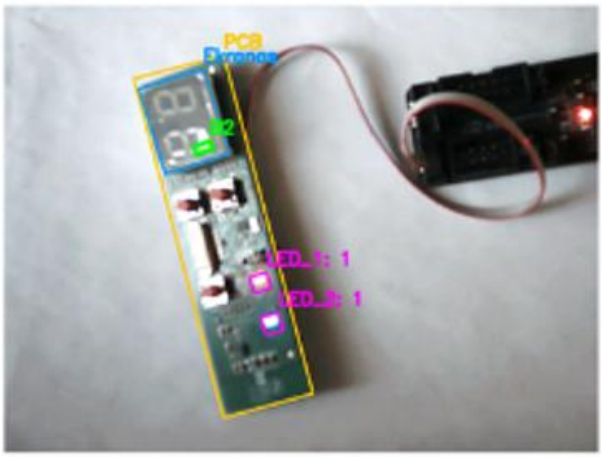
a) originalus kadras, kuriame segmentas nebuvo aptiktas



b) to paties kadro atpažinimo rezultatas; tikroji būsena – „B2“; nustatyta būsena – neaptikta



a) originalus kadras, kuriame segmentas buvo klaidingai priskirtas kitam segmentui



b) to paties kadro atpažinimo rezultatas; tikroji būsena – „A2“; nustatyta būsena – „B2“

25 pav. Klaidingo aktyvių segmentų atpažinimo pavyzdžiai

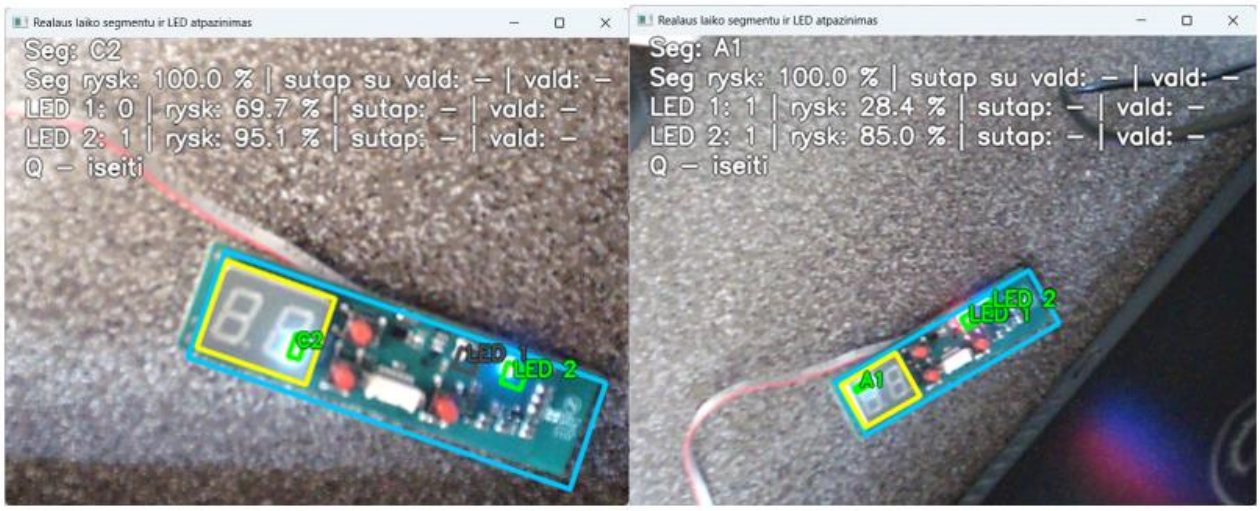
Šio projekto rezultatas – 99,17 proc. kadrų tikslumas, 99,89 proc. segmentų būsenų tikslumas ir 100 proc. LED būsenų tikslumas – yra artimas minėtų metodų rezultatams, tačiau tiesiogiai su jais nelyginamas kaip absoliučiai geresnis ar blogesnis, nes skiriasi tiriamieji objektai, duomenų rinkiniai, kameros sąlygos ir vertinimo metrikos. Šiame projekte vertintas ne defektų klasifikavimas ar skaitmens kaip simbolio atpažinimas, o konkrečių segmentinio ekrano ir LED „ON“ / „OFF“ būsenų sutapimas su valdiklio sugeneruota etalonine būsena. Dauguma nustatytų klaidų buvo susijusios ne su LED būsenų atpažinimu, o su aktyvaus segmento supainiojimu: iš 840 kadrų klaidingai atpažinti 7 kadrai, o abiejų LED būsenos visuose kadruose sutapo su etalonu.

5.7. Programos demonstravimas tiesioginiame kameros sraute

Tiesioginio kameros srauto programa realizuota „Python“ aplinkoje, naudojant „OpenCV“, „PyTorch“ ir „YOLOv5“ segmentavimo modelį, o pagrindinis YOLOv5 ir ROI pagrindu veikiančio būsenų atpažinimo programos kodas pateiktas 3 priede. Paleidimo metu įkeliamas apmokytas modelis, nuskaitytas ROI išdėstymo failas, inicijuojama USB kamera ir paruošiama serijinė jungtis su valdikliu. Šioje versijoje kameros numeris nustatytas kaip „CAMERA_ID = 1“, o valdiklio ryšiui

naudojama „COM4“ jungtis, kurios sparta yra 115 200 Bd. Programos paskirtis – be tarpinių failų saugojimo tiesiogiai iš kameros srauto nustatyti aktyvų ekrano segmentą ir dviejų papildomų LED būsenas bei rezultatą parodyti vartotojo lange.

Pirmajame tinkamame kadre atliekamas pilnas aptikimas. Tuo metu neuroninis tinklas ieško dviejų objektų klasių: spausdintinės plokštės („PCB_CLASS_ID = 0“) ir segmentinio ekrano srities („SEG_CLASS_ID = 1“). Iš gautų kaukių apskaičiuojami kontūrai ir keturkampiai, po to parenkama tokia PCB orientacija, kuri geriausiai atitinka etaloninį ROI išdėstymą. Remiantis šia orientacija, visi ROI iš etaloninės sistemos perkeliami į einamąjį kadrą homografijos būdu, o ekrano sritis papildomai patikslinama pagal faktiškai aptikto indikatoriaus padėtį. Taigi, kiekviename pilno aptikimo etape iš naujo apskaičiuojamos ekrano, segmentų ir LED zonos, o PCB kontūras atvaizdavimui gaunamas iš jau suderintų „SCREEN“, „LED_1“ ir „LED_2“ zonų padėties (žr. 26 pav.).



26 pav. Tiesioginio kameros srauto demonstravimo kadrai

Pilnas pakartotinis objektų aptikimas nevykdomas kiekviename kadre, kad programa veiktų sparčiau. Tarp pilnų aptikimų naudojamas optinis sekimas. Šiam tikslui iš ekrano ir LED sričių išrenkami kampiniai taškai, kurie sekami tarp gretimų kadrų naudojant optinio srauto metodą. Iš sekimo rezultatų apskaičiuojama afininė transformacija, kuri taikoma visiems anksčiau nustatytiems ROI. Pilnas aptikimas kartojamas pradžioje, kas 18 kadrų, po statiškumo režimo pabaigos arba tada, kai blogėja geometrijos tinkamumo rodikliai „fit_score“ ir „screen_iou“. 18 kadrų intervalas pasirinktas empiriškai kaip kompromisas tarp ROI stabilumo ir skaičiavimo apkrovos: dažnesnis aptikimas didino mirgėjimą ir apkrovą, o retesnis didino riziką, kad ROI vėluos paskui pasikeitusią geometriją. Jei optinis sekimas nepavyksta, naudojamas naujas pilnas aptikimas, o jei ir jis negrąžina tinkamo rezultato, išlaikoma paskutinė gera geometrija.

Papildomai programoje įgyvendinta statiškumo kontrolė. Kiekvienam naujam kadrui apskaičiuojamas skirtumas nuo ankstesnio sulieto pilko vaizdo. Jei keli kadrai iš eilės praktiškai nesikeičia, įjungiamas statiško vaizdo režimas, kuriame geometrija nebeperskaičiuojama, o išlaikomos paskutinės stabilios būsenos. Kai kadro pokytis vėl padidėja, programa grįžta į įprastą sekimo ir periodinio pakartotinio aptikimo režimą. Segmentų ir LED būsenoms taikomas laiko filtras, kuriame nauja reikšmė priimama tik po kelių vienodų kadrų iš eilės. Šioje versijoje patvirtinimui naudojami trys kadrai („STATE_CONFIRM_FRAMES = 3“). Šis skaičius pasirinktas empiriškai:

vienas kadras buvo per jautrus atsitiktiniams atspindžiams, o ilgesnis langas didino reakcijos vėlinimą.

Būsenų sprendimas atliekamas ROI pagrindu. Segmentų atveju iš kiekvieno ROI papildomai išskiriama vidinė vertinimo sritis, pritaikyta horizontaliems ir vertikaliems segmentams, ir joje apskaičiuojami spalviniai bei šviesumo požymiai. Sprendimas grindžiamas baltumo įverčiu, V kanalo ryškiu ir RGB kanalų sklaida. LED zonoms naudojami atskiri požymiai: raudonos arba mėlynos spalvos perteklius, sodrumas ir šviesumas. Apskaičiavus visas segmentų būsenas vizualizacijoje rodomas aktyvaus segmento pilnas ROI. LED 1 ir LED 2 rodomi pagal jų atitinkamų ROI būseną.

Galutinis rezultatas atvaizduojamas tiesiogiai kameros lange. Ekrane braižomi PCB ir ekrano rėmeliai, aktyvaus segmento ROI bei abiejų LED zonos. Tekstinėje dalyje rodoma aptikto segmento žyma, segmento ryškis procentais, LED 1 ir LED 2 būsenos bei jų ryškis. Programoje taip pat numatyta valdiklio būsenų nuskaitymo logika per UART: saugoma aptikta segmento žyma, LED būsenos ir galima skaičiuoti jų sutapimą su valdiklio siunčiamais duomenimis. Darbo ciklas vykdomas tol, kol vartotojas jį nutraukia paspausdamas klavišą „Q“.

Tiesioginio kameros srauto demonstravimas šiame darbe naudotas praktiniam algoritminės grandinės veikimui parodyti. Šis bandymas nebuvo laikomas atskiru kiekybiniu našumo eksperimentu, todėl FPS, vieno kadro apdorojimo trukmė, YOLO aptikimo trukmė, ROI analizės trukmė ir CPU / GPU apkrova atskirai nebuvo registruoti. Dėl šios priežasties 5.7 poskyris pagrindžia programos veikimo principą tiesioginiame kameros sraute, o ne pateikia pilną realaus laiko našumo įvertinimą. Galutinis algoritmo tikslumas vertintas 5.6 poskyryje pagal 840 kadrų funkcinio vertinimo imtį.

5.8. Apibendrinimas

Eksperimentinėje dalyje buvo įvertinta visa sukurta optinių indikatorių būsenų atpažinimo grandinė: YOLOv5 pagrįstas PCB ir segmentinio ekrano lokalizavimas, kadro geometrijos suvienodinimas, ROI sričių perkėlimas ir segmentų bei LED indikatorių būsenų nustatymas pagal šviesumo ir spalvinius požymius. YOLOv5 modelis šiame darbe naudotas ne galutinei indikatorių būsenai klasifikuoti, o tiriamojo objekto ir segmentinio ekrano padėčiai nustatyti, kad būsenos būtų vertinamos konkrečiose fizinėse ROI srityse.

Pirminio duomenų rinkimo metu sukauptas 6 840 kadrų rinkinys leido patikrinti sistemos veikimą, parengti lokalizavimo modelį ir sukonfigūruoti ROI pagrindu veikiančią būsenų nustatymo logiką. Galutiniame tyrimo etape skaitinių reikšmių kaip simbolių atpažinimo atsisakyta, todėl algoritmo tikslumas vertintas pagal 840 kadrų pavienių segmentų ir LED indikatorių būsenų imtį. Šioje imtyje klaidingai atpažinti 7 kadrai, bendras kadrų lygio tikslumas siekė 99,17 proc., segmentų „ON“ / „OFF“ būsenų tikslumas – 99,89 proc., o abiejų LED indikatorių būsenos nustatytos 100,00 proc. tikslumu.

Klaidų analizė parodė, kad galutinėje imtyje nepasitaikė techninių PCB ar segmentinio ekrano neaptikimo klaidų. Visi klaidingi atvejai buvo susiję su aktyvaus segmento nustatymu ir koncentravosi dešiniojo skaitmens srityje. Pagrindiniai tikslumą ribojė veiksniai buvo atspindžiai, lokalaus ryškumo iškraipymai ir kameros ekspozicijos prisitaikymas, keičiantys požymių reikšmes ROI srityse. Tai rodo, kad sukurta YOLOv5 ir ROI analizės grandinė yra tinkama tiriamojo objekto funkcinių būsenų nustatymui, tačiau praktiniam taikymui svarbu užtikrinti stabilesnes vaizdo fiksavimo sąlygas.

Išvados

1. Išanalizavus elektronikos gaminių optinių elementų testavimo metodus nustatyta, kad kontaktiniai, spektriniai ir specializuoti optiniai matavimai tinka tiksliems optiniams ar elektriniams parametrams vertinti, tačiau šio darbo uždaviniui jie yra pertekliniai. Segmentų ir LED „ON“ / „OFF“ būsenoms nustatyti tinkamesnis bekontaktis vaizdo analizės metodas.
2. Ištyrus vaizdų atpažinimo metodus nustatyta, kad vien slenkstinis apdorojimas yra jautrus apšvietimui, atspindžiams ir objekto padėčiai, o vien DNT klasifikatorius reikalautų didelio visų galimų būsenų duomenų rinkinio. Todėl pasirinktas mišrus YOLOv5 ir ROI analizės sprendimas: YOLOv5 lokalizuoja PCB ir segmentinio ekrano sritį, o segmentų ir LED būsenos nustatomos pagal ROI požymius.
3. Suprojektuota ir eksperimentiškai realizuota optinių indikatorių testavimo sistema, sudaryta iš segmentų ekrano, dviejų LED indikatorių, ESP32-S3 mikrovaldiklio, USB kameros ir kompiuterio. Pirminio duomenų rinkimo metu sukauptas 6 840 kadrų rinkinys, kuris naudotas sistemos veikimui patikrinti, lokalizavimo modeliui parengti ir ROI pagrindu veikiančiai būsenų nustatymo logikai sukonfigūruoti.
4. Sukurtas vaizdų apdorojimo algoritmas ir programinis sprendimas, kuriame PCB ir segmentinio ekrano sritys lokalizuojamos YOLOv5 modeliu, o segmentų ir LED būsenos nustatomos pagal ROI srityse apskaičiuotus šviesumo ir spalvinius požymius. YOLOv5 lokalizavimo modeliui sužymėti 75 vaizdai: 60 naudota mokymui, 15 – validacijai. Programos derinimui naudota 15 pavienių segmentų kadrų imtis.
5. Eksperimentiškai įvertinus algoritmą pagal 840 kadrų imtį nustatyta, kad klaidingai atpažinti 7 kadrai, todėl bendras kadrų lygio tikslumas siekė 99,17 proc. Segmentų „ON“ / „OFF“ būsenų tikslumas siekė 99,89 proc., o abiejų LED indikatorių būsenos nustatytos 100,00 proc. tikslumu. Klaidos buvo susijusios su aktyvaus segmento nustatymu dešiniojo skaitmens srityje, daugiausia dėl atspindžių, lokalių ryškumo pokyčių ir kameros ekspozicijos prisitaikymo.

Literatūros sąrašas

1. POPPE, A., HANTOS, G., HEGEDUS, J., CSUTI, P., RENCZ, M. Concepts for high throughput LED testing using high-speed optical transients of LEDs. *Therminic* [interaktyvus]. 2022, 1–7 [žiūrėta 2025-03-12]. Prieiga per: https://ieeexplore-ieee-org.ezproxy.ktu.edu/stamp/stamp.jsp?tp=&arnumber=9950671.
2. QIAO, J., JIN, Z., YANG, J., WANG, J., ZHANKUN, W., LI, J., GUO, R. Research on the. measurement system for interconnection bonding force of panel-level mikro-led chips. *SSRN* [interaktyvus]. 2025, 1–12 [žiūrėta 2025-04-12]. Prieiga per: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=5208379.
3. HENLEY, F. J. Light emitting diode (LED) test apparatus and method of manufacture. *United States Patent* [interaktyvus]. 2021, 1–39 [žiūrėta 2025-05-08]. Prieiga per: https://patents.google.com/patent/US10989755B2/en.
4. LIAO, CH., CHANG, T., LU, CH. Wafer-grade LED detection device and method. *United States Patent* [interaktyvus]. 2022, 1–11 [žiūrėta 2025-05-08]. Prieiga per: https://patents.google.com/patent/US11215525B2/en.
5. LING, Y. Method of building model of defect inspection for LED display. *United States Patent* [interaktyvus]. 2023, 1–11 [žiūrėta 2025-05-08]. Prieiga per: https://patents.google.com/patent/US11790509B2/en.
6. HUANG, X., YANG, Q. Flexible probe for mikroled defect detection and manufacturing method therefor. *United States Patent Application Publication* [interaktyvus]. 2024, 1–12 [žiūrėta 2025-05-08]. Prieiga per: https://patents.google.com/patent/US20240085493A1/en.
7. LEE, J. J., SCHUELE, P. J. System and method for the repair of serially connected display elements. *United States Patent* [interaktyvus]. 2024, 1–21 [žiūrėta 2025-04-20]. Prieiga per: https://patents.google.com/patent/US11916163B2/en.
8. YASUDA, T., SUGAWA, S., YOKOMICHI, Y., KOBAYASHI, K., HAMORI, H., TERAMOTO, A. High-speed and contactless inspection of defective mikro-LEDs through the photovoltaic effect. *Journal of the Society for Information Display* [interaktyvus]. 2024, 32, 825–835 [žiūrėta 2025-02-28]. Prieiga per: https://sid.onlinelibrary.wiley.com/doi/epdf/10.1002/jsid.2013.
9. JHA, S. B., BABICEANU, R. F. Deep CNN-based visual defect detection: Survey of current literature. *Computers in Industry* [interaktyvus]. 2023, 148, 1–14 [žiūrėta 2025-03-08]. Prieiga per: https://www-sciencedirect-com.ezproxy.ktu.edu/science/article/pii/S0166361523000611.
10. NURUL AFIFAH, A. N., INDRABAYU, SUYUTI, A., et. Al. A review on image processing techniques for damage detection on photovoltaic panels. *ICIC Express Letters. ICIC International* [interaktyvus]. 2021, 15, 779–790 [žiūrėta 2025-04-20]. Prieiga per: https://repository.unhas.ac.id/id/eprint/4945/1/A%20Review%20on%20Image%20Processing%20Techniques%20for%20Damage%20detection%20on%20Photovoltaic%20Panels.pdf.
11. SINGH, K., KHARCHE, S., CHAUHAN, A., SALVI, P. PCB defect detection methods: A review of existing methods and potential enhancements. *Journal of Engineering Science and Technology Review* [interaktyvus]. 2024, 17, 156–167 [žiūrėta 2025-04-20]. Prieiga per: https://www.researchgate.net/profile/Khushi-Singh-47/publication/PCB-Defect-Detection-Methods-A-Review-of-Existing-Methods-and-Potential-Enhancements.pdf.
12. LING, Q., MAT ISA, N. A. Printed circuit board defect detection methods based on image processing, machine learning and deep learning: A survey. *IEEE Access* [interaktyvus]. 2023, 11,

15921–15944 [žiūrėta 2025-04-20]. Prieiga per: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10044670.
13. ABU EBAYYEH, M. A. *Deep Learning for Automatic Optical Inspection and Quality Evaluation of Semiconductor and Optoelectronic Manufacturing* [interaktyvus]. 2022 [žiūrėta 2025-02-28]. Prieiga per: https://bura.brunel.ac.uk/handle/2438/24711.
14. ISLAM, M. R., ZAMIL, M. Z. H., RAYED, M. E., KABIR, M. M., MRIDHA, M. F., NISHIMURA, S., SHIN, J. Deep learning and computer vision techniques for enhanced quality control in manufacturing processes. *IEEE Access* [interaktyvus]. 2024, 12, 121449–121479 [žiūrėta 2025-03-06]. Prieiga per: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=10663422.
15. CHEN, X., WU, Y., HE, X., MING, W. A comprehensive review of deep learning-based PCB defect detection. *IEEE Access* [interaktyvus]. 2023, 11, 139017–139038 [žiūrėta 2025-04-20]. Prieiga per: https://ieeexplore.ieee.org/abstract/document/10343144.
16. FONSECA, L. A. L. O., IANO, Y., DE OLIVEIRA, G. G., VAZ, G. C., CARNIELLI, G. P., PEREIRA, J. C., ARTHUR, R. Automatic printed circuit board inspection: A comprehensible survey. *Discover Artificial Intelligence* [interaktyvus]. 2024, 4, 1–20 [žiūrėta 2025-04-20]. Prieiga per: https://link.springer.com/article/10.1007/s44163-023-00081-5.
17. KOH, E., KIM, H. D., BAEK, S., LEE, C. jouricial intelligence (AI) fundamentals for the display industry: A review. *IEEE Open Journal on Immersive Displays* [interaktyvus]. 2024, 1, 204–213 [žiūrėta 2025-03-06]. Prieiga per: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10700054.
18. OWEN, A., HANNAH, J. Automated quality assurance using AI and computer vision. *ResearchGate* [interaktyvus]. 2021, 2–46 [žiūrėta 2025-04-06]. Prieiga per: https://www.researchgate.net/profile/Antony-Owen/publication/Automated-Quality-Assurance-Using-AI-and-Computer-Vision.pdf.
19. HELDT, O. *Testing of LED Panels Using Computer Vision* [interaktyvus]. 2022, 1–36 [žiūrėta 2025-04-20]. Prieiga per: https://www.theseus.fi/bitstream/handle/10024/748279/Heldt_Otto.pdf?sequence=2.
20. LIU, M., IBRAHIM, M. S., WEN, M., LI, SH., WANG, A., ZHANG, G., FAN, J. Machine learning assisted early anomaly detection of LEDs with spectral power distribution modeling *IEEE* [interaktyvus]. 2023, 10, 185–189 [žiūrėta 2025-04-20]. Prieiga per: https://ieeexplore-ieee-org.ezproxy.ktu.edu/stamp/stamp.jsp?tp=&arnumber=10071010.
21. LI, Z., HUANG, M., CAO, X., XU, D., LIN, Y., CHEN, Z., GUO, Z. Proposing an accurate and fast optical batch inspection method of mini- / mikro-leds. *Journal of the Electron Devices Society* [interaktyvus]. 2024, 12, 289–295 [žiūrėta 2025-02-28]. Prieiga per: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10495305.
22. PARK, S., KO, J. H. Robust inspection of mikro-led chip defects using unsupervised anomaly detection chips. *ICTC* [interaktyvus]. 2021, 978, 1841–1843 [žiūrėta 2025-04-20]. Prieiga per: https://ieeexplore-ieee-org.ezproxy.ktu.edu/stamp/stamp.jsp?tp=&arnumber=9620801.
23. WANG, Y., CHU, J., CHEN, Y., LIANG, D., WEN, K., CAI, J. Dual entropy-controlled convolutional neural network for mini / mikro LED defect recognition. *IEEE Transactions on Instrumentation and Measurement* [interaktyvus]. 2023, 72, 1–14 [žiūrėta 2025-04-20]. Prieiga per: https://ieeexplore-ieee-org.ezproxy.ktu.edu/stamp/stamp.jsp?tp=&arnumber=10297995&tag=1.
24. LIN, Y., PAN, SH., YU, J., HONG, Y., WANG, F., ZHENG, L., TANG, J., CHEN, S. Mdca-detr: DETR with multi-channel deformable convolution and coordinate attention for mini-LED

wafer surface defects detection. *Optics and Lasers in Engineering* [interaktyvus]. 2024, 10, 185–189 [žiūrėta 2025-04-20]. Prieiga per: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=5043726.
25. SHU, Y., LI, B., LIN, H. Quality safety monitoring of LED chips using deep learning-based vision inspection methods. *Measurement* [interaktyvus]. 2021, 168, 1–10 [žiūrėta 2025-03-08]. Prieiga per: https://www-sciencedirect-com.ezproxy.ktu.edu/science/article/pii/S0263224120306618.
26. KO, S. J., KIM, J. W., WOO, J. S., HONG, S. J., KIM, G. Image processing and deep learning-based defect detection theory for sapphire epi-wafer in green LED manufacturing. *Journal of the Semiconductor & Display Technology* [interaktyvus]. 2024, 1–21 [žiūrėta 2025-04-20]. Prieiga per: https://koreascience.kr/article/JAKO202320857535222.pdf.
27. ZHOU, Y., WANG, W., ZENG, D., HUANG, G., YU, C. A cascaded mini / mikro LED defect detection method based on efficient unsupervised segmentation network. *Computers in Industry* [interaktyvus]. 2024, 1–16 [žiūrėta 2025-04-20]. Prieiga per: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=5031157.
28. ZHANG, G., CUI, K., HUNG, T. Y., LU, S. Defect-GAN: High-fidelity defect synthesis for automated defect inspection. *Computer Vision Foundation* [interaktyvus]. 2021, 2524–2534 [žiūrėta 2025-04-06]. Prieiga per: https://openaccess.thecvf.com/content/WACV2021/html/Zhang_Defect-GAN_High-Fidelity_Defect_Synthesis_for_Automated_Defect_Inspection_WACV_2021_paper.html.
29. XU, L., HU, X., HE, T., HU, K., ZHANG, J. Defect detection on LED chips based on position pre-estimation and feature enhancement. *Applied Sciences* [interaktyvus]. 2022, 12, 1–12 [žiūrėta 2025-04-06]. Prieiga per: https://www.mdpi.com/2076-3417/12/3/1265.
30. ZHENG, P., LOU, J., WAN. X., LUO, Q., LI, Y., XIE, L., ZHU, Z. LED chip defect detection method based on a hybrid algorithm. *International Journal of Intelligent Systems* [interaktyvus]. 2023, 1–13 [žiūrėta 2025-03-06]. Prieiga per: https://onlinelibrary.wiley.com/doi/epdf/10.1155/2023/4096164.
31. CHU, J., CAI, J., LI, L., HOU, B., WEN, K., LIANG, D. Small dense mini / mikro LED high-precision inspection based on instance segmentation with local detail enhancement. *Advanced Engineering Informatics* [interaktyvus]. 2025, 65, 1–11 [žiūrėta 2025-03-12]. Prieiga per: https://www-sciencedirect-com.ezproxy.ktu.edu/science/article/pii/S1474034625000928.
32. CHEN, M., HAN, SH., LI, CH. Efficient Mikro-LED defect detection based on mikroscopic vision and deep learning. *Optics and Lasers in Engineering* [interaktyvus]. 2024, 4, 1–20 [žiūrėta 2025-04-20]. Prieiga per: https://www-sciencedirect-com.ezproxy.ktu.edu/science/article/pii/S0143816624000952.
33. CHEN, S., H., TSAI, C. C. SMD LED chips defect detection using a YOLOV3-dense model. *Advanced Engineering Informatics* [interaktyvus]. 2021, 47, 1–14 [žiūrėta 2025-04-20]. Prieiga per: https://www-sciencedirect-com.ezproxy.ktu.edu/science/article/pii/S1474034621000100.
34. ZHONG, Z., LI, C., CHEN, M., WU, H., KIYOSHI, T. Mikro LED defect detection with self-attention mechanism-based neural network. *Digital Signal Processing* [interaktyvus]. 2024, 149, 1–12 [žiūrėta 2025-04-06]. Prieiga per: https://www-sciencedirect-com.ezproxy.ktu.edu/science/article/pii/S105120042400099X.
35. SUN, X., SHI, W., CHENG, Q., LIU, W., WANG, Z., ZHANG, J. An LED detection and recognition method based on deep learning in vehicle optical camera communication. *IEEE Access* [interaktyvus]. 2021, 9, 80897–80905 [žiūrėta 2025-04-20]. Prieiga per: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9444430.

36. ZHU, H., HUANG, J., LIU, H., ZHOU, Q., ZHU, J., LI, B. Deep-learning-enabled automatic optical inspection for module-level defects in LCD. *IEEE Internet of Things Journal* [interaktyvus]. 2022, 9, 1122–1135 [žiūrėta 2025-04-20]. Prieiga per: https://ieeexplore-ieee-org.ezproxy.ktu.edu/stamp/stamp.jsp?tp=&arnumber=9429707.
37. ZHAO, G., ZHENG, X., HUANG, X., LU, Y., CHEN, Z., GUO, W. Vehicular Mini-LED backlight display inspection based on residual global context mechanism. *Frontiers of Optoelectronics* [interaktyvus]. 2024, 17, 1–10 [žiūrėta 2025-04-20]. Prieiga per: https://link.springer.com/article/10.1007/s12200-024-00140-4.
38. STERN, M., L., SCHELLENBERGER, M. Fully convolutional networks for chip-wise defect detection employing photoluminescence images. *Journal of Intelligent Manufacturing* [interaktyvus]. 2021, 32, 113–126 [žiūrėta 2025-04-20]. Prieiga per: https://link-springer-com.ezproxy.ktu.edu/article/10.1007/s10845-020-01563-4.
39. GEMAR, M., D., PAN, SH., ZHANG, ZH., MACHEMEH, R. B. Fuzzy reliability theory analysis of traffic signal lamp performance. *Multimodal Transportation* [interaktyvus]. 2025, 4, 1–9 [žiūrėta 2025-05-08]. Prieiga per: https://www.sciencedirect.com/science/article/pii/S2772586325000097.
40. CHEN, M., CHEN, J., LI, CH., WANG, Q., TAKAMASU, K. Defect detection of mikro-LED with low distinction based on deep learning. *Optics and Lasers in Engineering* [interaktyvus]. 2024, 173, 1–9 [žiūrėta 2025-05-08]. Prieiga per: https://www-sciencedirect-com.ezproxy.ktu.edu/science/article/pii/S0143816623004530.
41. *Kompiuteris „MSI Thin 15 B13VE“* [interaktyvus]. 2025 [žiūrėta 2025-12-20]. Prieiga per: https://www.msi.com/Laptop/Thin-15-B13VX/Specification
42. *Logitech QuickCam Communicate MP USB kamera* [interaktyvus]. 2025 [žiūrėta 2025-12-20]. Prieiga per: https://www.bhphotovideo.com/c/product/561894-REG/Logitech_960_000240_QuickCam_Communicate_MP_USB.html/specs
43. *ESP32-S3 mikrovaldiklis* [interaktyvus]. 2025 [žiūrėta 2025-12-20]. Prieiga per: https://documentation.espressif.com/esp32-s3_datasheet_en.pdf
44. *ESP-Prog programatorius* [interaktyvus]. 2025 [žiūrėta 2025-12-20]. Prieiga per: https://docs.espressif.com/projects/esp-dev-kits/en/latest/other/esp-dev-kits-en-master-other.pdf
45. REN, Z., FANG, F., YAN, N., WU, Y. State of the art in defect detection based on machine vision. *International Journal of Precision Engineering and Manufacturing-Green Technology* [interaktyvus]. 2022, 9, 661–691 [žiūrėta 2026-03-18]. Prieiga per: https://link.springer.com/article/10.1007/s40684-021-00343-6.
46. MARUSCHAK, P., KONOVALENKO, I., OSADTSA, Y., MEDVID, V., SHOVKUN, O., BARAN, D., KOZBUR, H., MYKHAILYSHYN, R. Surface illumination as a factor influencing the efficacy of defect recognition on a rolled metal surface using a deep neural network. *Applied Sciences* [interaktyvus]. 2024, 14, 2591 [žiūrėta 2026-03-18]. Prieiga per: https://www.mdpi.com/2076-3417/14/6/2591.
47. WANNACHAI, A., BOONYUNG, W., CHAMPRASERT, P. real-time seven segment display detection and recognition online system using CNN. *Bio-inspired Information and Communication Technologies* (BICT 2020) [interaktyvus]. 2020, 52–67 [žiūrėta 2026-03-18]. Prieiga per: https://link-springer-com.ezproxy.ktu.edu/chapter/10.1007/978-3-030-57115-3_5.

Priedai

1 priedas. ESP32-S3 valdiklio programos kodas

```
#include <Preferences.h>
#include "esp_timer.h"
//PINAI
const int PIN_SEG_A = 35;
const int PIN_SEG_B = 36;
const int PIN_SEG_C = 37;
const int PIN_SEG_D = 38;
const int PIN_SEG_E = 39;
const int PIN_SEG_F = 40;
const int PIN_SEG_G = 41;
const int PIN_DIG1 = 47;
const int PIN_DIG2 = 48;
const int PIN_BTN_UP   = 11;
const int PIN_BTN_DOWN = 12;
const int PIN_BTN_LED  = 13;
const int PIN_LED_1 = 17;
const int PIN_LED_2 = 18;
inline int segActive()   { return LOW;  }
inline int segInactive() { return HIGH; }
inline int digActive()   { return LOW;  }
inline int digInactive() { return HIGH; }
const uint32_t MUX_US = 900;
const uint32_t DEBOUNCE_MS = 25;
const uint32_t REPEAT_DELAY_MS = 500;
const uint32_t REPEAT_RATE_MS  = 60;
const uint32_t SAVE_IDLE_MS    = 1000;
const uint8_t FADE_STEPS = 8;
Preferences prefs;
const uint8_t DIGIT_LUT[10] = {
  0b0111111, 0b0000110, 0b1011011, 0b1001111, 0b1100110,
  0b1101101, 0b1111101, 0b0000111, 0b1111111, 0b1101111
};
const int segPins[7] = {
  PIN_SEG_A, PIN_SEG_B, PIN_SEG_C, PIN_SEG_D,
  PIN_SEG_E, PIN_SEG_F, PIN_SEG_G
};
//Mygtukai
struct Btn {
  int pin;
  bool lastStable = HIGH;
  bool lastRead   = HIGH;
  uint32_t lastChange = 0;
  uint32_t pressedAt  = 0;
  uint32_t lastRpt    = 0;
  bool repeating = false;
};
Btn btnUp{PIN_BTN_UP}, btnDn{PIN_BTN_DOWN}, btnLed{PIN_BTN_LED};
volatile int currentValue = 0;
volatile int prevValue    = 0;
volatile uint8_t fadeLevel = FADE_STEPS;
volatile uint8_t ledMode = 0;
volatile bool driveLeft = true;
enum DisplayMode : uint8_t { MODE_NORMAL = 0, MODE_WALK = 1 };
volatile DisplayMode dispMode = MODE_NORMAL;
volatile uint8_t walkDigit = 0;  // 0=kairys, 1=dešinys
volatile uint8_t walkSeg   = 0;  // 0..6 = a..g
String uartBuffer = "";
portMUX_TYPE lockMux = portMUX_INITIALIZER_UNLOCKED;
esp_timer_handle_t muxTimer;
//MULTIPLEX ISR
void muxTimerCB(void*) {
  if (dispMode == MODE_WALK) {
    digitalWrite(PIN_DIG1, digInactive());
    digitalWrite(PIN_DIG2, digInactive());
    uint8_t pat = (1u << walkSeg);
```

```
    for (int i = 0; i < 7; i++)
      digitalWrite(segPins[i], ((pat >> i) & 1) ? segActive() : segInactive());
    if (walkDigit == 0)
      digitalWrite(PIN_DIG1, digActive());
    else
      digitalWrite(PIN_DIG2, digActive());
    return;
  }
  // Normalus režimas
  int val, prev;
  uint8_t fade;
  portENTER_CRITICAL(&lockMux);
  val  = currentValue;
  prev = prevValue;
  fade = fadeLevel;
  portEXIT_CRITICAL(&lockMux);
  if (val < 0) val = 0;
  if (val > 99) val = 99;
  int Lnew = val / 10;
  int Rnew = val % 10;
  int Lold = prev / 10;
  int Rold = prev % 10;
  digitalWrite(PIN_DIG1, digInactive());
  digitalWrite(PIN_DIG2, digInactive());
  bool left = driveLeft;
  static uint8_t slice = 0;
  uint8_t pat;
  if (fade < FADE_STEPS) {
    slice = (slice + 1) % FADE_STEPS;
    bool show = (slice < fade);
    int d = left ? (show ? Lnew : Lold) : (show ? Rnew : Rold);
    pat = DIGIT_LUT[d];
  } else {
    int d = left ? Lnew : Rnew;
    pat = DIGIT_LUT[d];
  }
  for (int i = 0; i < 7; i++)
    digitalWrite(segPins[i], ((pat >> i) & 1) ? segActive() : segInactive());
  digitalWrite(left ? PIN_DIG1 : PIN_DIG2, digActive());
  driveLeft = !driveLeft;
}
//Mygtukų funkcija
bool handleBtn(Btn &b, uint32_t now);
bool handleBtn(Btn &b, uint32_t now) {
  bool raw = digitalRead(b.pin);
  if (raw != b.lastRead) {
    b.lastRead = raw;
    b.lastChange = now;
  }
  if ((now - b.lastChange) >= DEBOUNCE_MS && raw != b.lastStable) {
    b.lastStable = raw;
    if (raw == LOW) {
      b.pressedAt = now;
      b.repeating = false;
      b.lastRpt = now;
      return true;
    }
  }
  if (b.lastStable == LOW) {
    if (!b.repeating && (now - b.pressedAt) >= REPEAT_DELAY_MS) {
      b.repeating = true;
      b.lastRpt = now;
      return true;
    }
    if (b.repeating && (now - b.lastRpt) >= REPEAT_RATE_MS) {
      b.lastRpt = now;
      return true;
    }
```

```
  }
  return false;
}
//UART KOMANDOS
void processUartCommand(String cmd) {
  cmd.trim();
  // SETSTATE 0..113
  if (cmd.startsWith("SETSTATE")) {
    int id = cmd.substring(8).toInt();
    id = constrain(id, 0, 113);
    if (id <= 99) {
      portENTER_CRITICAL(&lockMux);
      prevValue = currentValue;
      currentValue = id;
      fadeLevel = 0;
      dispMode = MODE_NORMAL;
      portEXIT_CRITICAL(&lockMux);
    } else {
      int k = id - 100;          // 0..13
      uint8_t d = k / 7;         // 0..1
      uint8_t g = k % 7;         // 0..6
      portENTER_CRITICAL(&lockMux);
      dispMode = MODE_WALK;
      walkDigit = d;
      walkSeg = g;
      portEXIT_CRITICAL(&lockMux);
    }
    return;
  }
  if (cmd.startsWith("SETVAL")) {
    int v = cmd.substring(6).toInt();
    v = constrain(v, 0, 99);
    portENTER_CRITICAL(&lockMux);
    prevValue = currentValue;
    currentValue = v;
    fadeLevel = 0;
    dispMode = MODE_NORMAL;
    portEXIT_CRITICAL(&lockMux);
    return;
  }
  if (cmd.startsWith("SETLED")) {
    int m = cmd.substring(6).toInt();
    m &= 3;
    ledMode = m;
    digitalWrite(PIN_LED_1, (m == 1 || m == 2));
    digitalWrite(PIN_LED_2, (m == 2 || m == 3));
    return;
  }
}
//SETUP
void setup() {
  Serial.begin(115200);
  for (int p : segPins) {
    pinMode(p, OUTPUT);
    digitalWrite(p, segInactive());
  }
  pinMode(PIN_DIG1, OUTPUT);
  pinMode(PIN_DIG2, OUTPUT);
  pinMode(btnUp.pin, INPUT_PULLUP);
  pinMode(btnDn.pin, INPUT_PULLUP);
  pinMode(btnLed.pin, INPUT_PULLUP);
  pinMode(PIN_LED_1, OUTPUT);
  pinMode(PIN_LED_2, OUTPUT);
  prefs.begin("counter", false);
  currentValue = prevValue = prefs.getInt("val", 0);
  ledMode = prefs.getInt("ledMode", 0) & 3;
  prefs.end();
  digitalWrite(PIN_LED_1, (ledMode == 1 || ledMode == 2));
```

```
  digitalWrite(PIN_LED_2, (ledMode == 2 || ledMode == 3));
  esp_timer_create_args_t args = {};
  args.callback = &muxTimerCB;
  args.name = "mux";
  esp_timer_create(&args, &muxTimer);
  esp_timer_start_periodic(muxTimer, MUX_US);
  fadeLevel = FADE_STEPS;
}
//LOOP
void loop() {
  uint32_t now = millis();
  static uint32_t lastInput = 0;
  static uint32_t lastFade = 0;
  //UART
  while (Serial.available()) {
    char c = Serial.read();
    if (c == '\n' || c == '\r') {
      if (uartBuffer.length()) {
        processUartCommand(uartBuffer);
        uartBuffer = "";
      }
    } else {
      uartBuffer += c;
    }
  }
  //Mygtukai
  bool up  = handleBtn(btnUp, now);
  bool dn  = handleBtn(btnDn, now);
  bool led = handleBtn(btnLed, now);
  if (led) {
    ledMode = (ledMode + 1) & 3;
    digitalWrite(PIN_LED_1, (ledMode == 1 || ledMode == 2));
    digitalWrite(PIN_LED_2, (ledMode == 2 || ledMode == 3));
    lastInput = now;
  }
  if (up) {
    if (dispMode == MODE_NORMAL) {
      if (currentValue < 99) {
        portENTER_CRITICAL(&lockMux);
        prevValue = currentValue;
        currentValue++;
        fadeLevel = 0;
        portEXIT_CRITICAL(&lockMux);
      } else {
        // Start forward walk
        dispMode = MODE_WALK;
        walkDigit = 0;
        walkSeg = 0;
      }
    } else {
      if (++walkSeg >= 7) {
        walkSeg = 0;
        if (++walkDigit >= 2) {
          dispMode = MODE_NORMAL;
          currentValue = 0;
        }
      }
    }
    lastInput = now;
  }
  if (dn) {
    if (dispMode == MODE_NORMAL) {
      if (currentValue > 0) {
        portENTER_CRITICAL(&lockMux);
        prevValue = currentValue;
        currentValue--;
        fadeLevel = 0;
        portEXIT_CRITICAL(&lockMux);
```

```
    } else {
     // Start backward walk
     dispMode = MODE_WALK;
     walkDigit = 1;
     walkSeg = 6;
    }
   } else {
    if (walkSeg == 0) {
     walkSeg = 6;
     if (walkDigit == 0) {
      dispMode = MODE_NORMAL;
      currentValue = 99;
     } else {
      walkDigit = 0;
     }
    } else {
     walkSeg--;
    }
   }
   lastInput = now;
  }
  // Fade progresas
  if (FADE_STEPS && (now - lastFade) >= 12) {
   lastFade = now;
   if (fadeLevel < FADE_STEPS) fadeLevel++;
  }
  // Saugojimas
  static int lastSavedVal = currentValue;
  static int lastSavedLed = ledMode;
  if ((now - lastInput) >= SAVE_IDLE_MS) {
   prefs.begin("counter", false);
   prefs.putInt("val", currentValue);
   prefs.putInt("ledMode", ledMode);
   prefs.end();
   lastSavedVal = currentValue;
   lastSavedLed = ledMode;
  }
  delay(1);
}
```

2 priedas. Kadrų surinkimo ir UART ryšio programos kodas

```
import os
import csv
import time
import json
import datetime as dt
from dataclasses import dataclass
from typing import List, Optional, Tuple, Dict, Any
import cv2
import numpy as np
import serial
# Konfigūracija
BASE_DIR = r"C:\Python"
CAMERA_ID = 1
CAM_BACKEND = cv2.CAP_DSHOW
FRAME_W = 640
FRAME_H = 480
FPS = 30
SERIAL_PORT = "COM4"   # <-- pakeisk į tikrą
BAUD = 115200
SETTLE_MS = 200
POST_MS = 400
SERIAL_CMD_DELAY_MS = 20
LED_TAG = {0: "00", 1: "10", 2: "11", 3: "01"}
# Kamera po START turi susireguliuoti (auto exposure / gain)
CAM_WARMUP_S = 10.0
# Būsenos
@dataclass(frozen=True)
```

```
class State:
    kind: str
    tag: str
    val: Optional[int] = None
    seg_digit: Optional[int] = None
    seg_index: Optional[int] = None
def build_states() -> List[State]:
    states: List[State] = []
    for v in range(0, 100):
        states.append(State(kind="NUM", tag=f"val{v:02d}", val=v))
    for digit in (0, 1):
        for seg in range(0, 7):
            states.append(State(kind="SEG", tag=f"segD{digit}_S{seg}", seg_digit=digit, seg_index=seg))
    return states
def build_state_led_list(states: List[State]) -> List[Tuple[State, int]]:
    out: List[Tuple[State, int]] = []
    for st in states:
        for led_mode in (0, 1, 2, 3):
            out.append((st, led_mode))
    return out
# Katalogas + logai
def make_session_dir(base_dir: str, suggested_val: str = "val00") -> str:
    now = dt.datetime.now()
    stamp = now.strftime("%y%m%d_%H%M%S")
    prefix = f"{stamp}_{suggested_val}"
    idx = 1
    while True:
        name = f"{prefix}_S{idx:02d}"
        path = os.path.join(base_dir, name)
        if not os.path.exists(path):
            os.makedirs(path)
            return path
        idx += 1
def open_log(session_dir: str) -> Tuple[csv.writer, Any]:
    log_path = os.path.join(session_dir, "log.csv")
    f = open(log_path, "w", newline="", encoding="utf-8")
    w = csv.writer(f, delimiter=";")
    w.writerow([
        "ts_iso", "frame_idx",
        "state_kind", "state_tag", "val", "seg_digit", "seg_index",
        "led_mode", "led_tag",
        "filename",
        "analysis_ok", "error_code", "confidence", "notes"
    ])
    return w, f
def save_session_meta(session_dir: str, meta: Dict[str, Any]) -> None:
    p = os.path.join(session_dir, "session_meta.json")
    with open(p, "w", encoding="utf-8") as f:
        json.dump(meta, f, ensure_ascii=False, indent=2)
# UART + kamera
def uart_open() -> serial.Serial:
    ser = serial.Serial(SERIAL_PORT, BAUD, timeout=1)
    time.sleep(0.8)
    return ser
def uart_send(ser: serial.Serial, line: str) -> None:
    ser.write((line.strip() + "\n").encode("utf-8"))
    time.sleep(SERIAL_CMD_DELAY_MS / 1000.0)
def camera_open() -> cv2.VideoCapture:
    cam = cv2.VideoCapture(CAMERA_ID, CAM_BACKEND)
    if not cam.isOpened():
        raise RuntimeError(f"Nepavyko atidaryti kameros (CAMERA_ID={CAMERA_ID}).")
    cam.set(cv2.CAP_PROP_FRAME_WIDTH, FRAME_W)
    cam.set(cv2.CAP_PROP_FRAME_HEIGHT, FRAME_H)
    cam.set(cv2.CAP_PROP_FPS, FPS)
    # 10 s warm-up po START
    t0 = time.time()
    while time.time() - t0 < CAM_WARMUP_S:
        cam.read()
```

```
        time.sleep(0.02)
    for _ in range(10):
        cam.read()
    return cam
def camera_grab_one(cam: cv2.VideoCapture) -> np.ndarray:
    for _ in range(3):
        cam.read()
    ok, frame = cam.read()
    if not ok or frame is None:
        raise RuntimeError("Kamera neatidavė kadro.")
    return frame
# Analizė (karkasas)
def analysis_pipeline(_frame_bgr: np.ndarray, _expected: Tuple[State, int]) -> Tuple[bool, str, float, str]:
    return False, "ANALYSIS_NOT_IMPLEMENTED", 0.0, "Pipeline skeleton only"
# Failo vardas
def make_filename(session_dir: str, yymmdd: str, st: State, led_mode: int, frame_idx: int) -> str:
    led_tag = LED_TAG.get(led_mode, f"{led_mode:02d}")
    state_part = f"val{st.val:02d}" if st.kind == "NUM" else st.tag
    fname = f"{yymmdd}_{state_part}_led{led_tag}_{frame_idx:04d}.png"
    return os.path.join(session_dir, fname)
# Diagnostika prieš START
def preflight_checks() -> None:
    # BASE_DIR rašymas
    if not os.path.exists(BASE_DIR):
        raise RuntimeError(f"BASE_DIR neegzistuoja: {BASE_DIR}")
    test_dir = os.path.join(BASE_DIR, "_write_test")
    try:
        os.makedirs(test_dir, exist_ok=True)
        test_file = os.path.join(test_dir, "test.txt")
        with open(test_file, "w", encoding="utf-8") as f:
            f.write("ok")
        os.remove(test_file)
        os.rmdir(test_dir)
    except Exception as e:
        raise RuntimeError(f"Negalima rašyti į BASE_DIR: {e}")
    # Kamera
    cam = cv2.VideoCapture(CAMERA_ID, CAM_BACKEND)
    if not cam.isOpened():
        raise RuntimeError(f"Kamera neatsidaro (CAMERA_ID={CAMERA_ID}). Pakeiskite į 0/1/2.")
    cam.release()
    # UART
    try:
        ser = serial.Serial(SERIAL_PORT, BAUD, timeout=1)
        ser.close()
    except Exception as e:
        raise RuntimeError(f"UART neatsidaro (SERIAL_PORT={SERIAL_PORT}). Klaida: {e}")
# Seanso vykdymas
def run_session() -> None:
    print("1) Paruoškite PCB padėtį. Kai paruošta – spauskite ENTER (START).")
    input()
    # Inicializacija
    print("Atliekama inicializacinė patikra (katalogas / kamera / UART)...")
    preflight_checks()
    print("Inicializacija OK")
    # 2) Katalogas
    states = build_states()
    session_dir = make_session_dir(BASE_DIR, suggested_val="val00")
    print(f"2) Seanso katalogas: {session_dir}")
    # 3) Resursai
    print("3.1) UART...")
    ser = uart_open()
    print("3.2) Kamera...")
    cam = camera_open()
    meta = {
        "created_iso": dt.datetime.now().isoformat(timespec="seconds"),
        "base_dir": BASE_DIR,
        "camera": {"id": CAMERA_ID, "width": FRAME_W, "height": FRAME_H, "fps": FPS},
        "uart": {"port": SERIAL_PORT, "baud": BAUD},
```

```
    "timing_ms": {"settle_ms": SETTLE_MS, "post_ms": POST_MS},
    "counts": {"states": len(states), "total_items": len(states) * 4}
}
save_session_meta(session_dir, meta)
log_writer, log_file = open_log(session_dir)
items = build_state_led_list(states)
if len(items) != 456:
    raise RuntimeError(f"Tikėtasi 456 elementų, bet yra {len(items)}")
yymmdd = dt.datetime.now().strftime("%y%m%d")
stats = {"ok": 0, "fail": 0, "error_codes": {}}
try:
    for frame_idx, (st, led_mode) in enumerate(items, start=1):
        # 5.1
        if st.kind == "NUM":
            # 0..99
            uart_send(ser, f"SETSTATE {st.val}")
        else:
            # 100..113 (14 simbolių)
            sid = 100 + (st.seg_digit * 7 + st.seg_index)
            uart_send(ser, f"SETSTATE {sid}")
        uart_send(ser, f"SETLED {led_mode}")
        # 5.2
        time.sleep(SETTLE_MS / 1000.0)
        # 5.3
        frame = camera_grab_one(cam)
        # 5.4
        filepath = make_filename(session_dir, yymmdd, st, led_mode, frame_idx)
        ok_write = cv2.imwrite(filepath, frame)
        if not ok_write:
            raise RuntimeError(f"Nepavyko išsaugoti failo: {filepath}")
        # 5.6
        a_ok, err_code, conf, notes = analysis_pipeline(frame, (st, led_mode))
        # 5.5
        ts_iso = dt.datetime.now().isoformat(timespec="milliseconds")
        led_tag = LED_TAG.get(led_mode, f"{led_mode:02d}")
        log_writer.writerow([
            ts_iso, frame_idx,
            st.kind, st.tag,
            st.val if st.kind == "NUM" else "",
            st.seg_digit if st.kind == "SEG" else "",
            st.seg_index if st.kind == "SEG" else "",
            led_mode, led_tag,
            os.path.basename(filepath),
            1 if a_ok else 0, err_code, f"{conf:.3f}", notes
        ])
        if a_ok:
            stats["ok"] += 1
        else:
            stats["fail"] += 1
            stats["error_codes"][err_code] = stats["error_codes"].get(err_code, 0) + 1
        # 5.6.9
        time.sleep(POST_MS / 1000.0)
    summary = {
        "total": len(items),
        "ok": stats["ok"],
        "fail": stats["fail"],
        "error_codes": dict(sorted(stats["error_codes"].items(), key=lambda x: -x[1]))
    }
    with open(os.path.join(session_dir, "summary.json"), "w", encoding="utf-8") as f:
        json.dump(summary, f, ensure_ascii=False, indent=2)
    print("6) Seansas baigtas")
    print(f"- total: {summary['total']}, ok: {summary['ok']}, fail: {summary['fail']}")
finally:
    try:
        log_file.flush()
        log_file.close()
    except Exception:
        pass
```

```
        try:
            cam.release()
        except Exception:
            pass
        try:
            ser.close()
        except Exception:
            pass
def main():
    while True:
        try:
            run_session()
        except Exception as e:
            print("\nSTOP: klaida po START")
            print(e)
            print("Pataisyti konfigūraciją (COM / CAMERA_ID / BASE_DIR) ir bandyti vėl. \n")
            input("Spausti ENTER, kad neišmestų lango...")
            input("Spausti ENTER, kad tęsti...")
        print("7) Pakartotinis seansas? (y/n)")
        ans = input().strip().lower()
        if ans != "y":
            break
if __name__ == "__main__":
    main()
```

3 priedas. YOLOv5 ir ROI pagrindu veikiančio būsenų atpažinimo programos kodas

```
import sys
import os
import json
import time
from dataclasses import dataclass
import cv2
import numpy as np
import torch
try:
    import serial
except Exception:
    serial = None
# KELIAI
ROOT_DIR = r"C:\ Python\Tinkami"
YOLOV5_DIR = r"C:\Python\yolov5"
MODEL_PATH = r"C:\ Python\yolov5\runs\train-seg\exp\weights\best.pt"
ROI_LAYOUT_PATH = os.path.join(ROOT_DIR, "roi_layout.json")
# PARAMETRAI
CAMERA_ID = 1
IMG_SIZE = 640
CONF_THRES = 0.25
IOU_THRES = 0.45
PCB_CLASS_ID = 0
SEG_CLASS_ID = 1
DETECT_EVERY_N_FRAMES = 18
MIN_FIT_SCORE = 0.80
MIN_SCREEN_IOU = 0.18
STATIC_DIFF_THRESHOLD = 1.15
STATIC_CONSEC_FRAMES_TO_LOCK = 4
UNLOCK_MOTION_THRESHOLD = 2.30
BLUR_FOR_MOTION = (9, 9)
FLOW_MAX_CORNERS = 180
FLOW_QUALITY_LEVEL = 0.01
FLOW_MIN_DISTANCE = 7
FLOW_BLOCK_SIZE = 7
FLOW_WIN_SIZE = (21, 21)
FLOW_MAX_LEVEL = 3
FLOW_MIN_GOOD_POINTS = 10
FLOW_MAX_ERR = 25.0
FLOW_INLIER_RANSAC = 3.0
STATE_CONFIRM_FRAMES = 3
SERIAL_PORT = "COM4"
```

```
SERIAL_BAUD = 115200
SERIAL_TIMEOUT = 0
SERIAL_OPEN_WAIT_S = 1.8
SEGMENT_NAMES = {
    "SCREEN",
    "A1", "B1", "C1", "D1", "E1", "F1", "G1",
    "A2", "B2", "C2", "D2", "E2", "F2", "G2"
}
LED_NAMES = {"LED_1", "LED_2"}
MIN_LED_SAT = 20.0
LED_TOP_BRIGHT_FRAC = 0.20
LED_MIN_PIXELS = 12
HSEG_X1, HSEG_X2 = 2.0 / 6.0, 4.0 / 6.0
HSEG_Y1, HSEG_Y2 = 0.0, 1.0
VSEG_X1, VSEG_X2 = 0.0, 1.0
VSEG_Y1, VSEG_Y2 = 2.0 / 6.0, 4.0 / 6.0
SEG_WHITE_MIN = 250.0
SEG_V_MIN = 255.0
SEG_SPREAD_MAX = 10.0
DIGIT_SEGMENTS = ["A", "B", "C", "D", "E", "F", "G"]
CENTER_NEIGHBORS = {
    "A": ("F", "B"),
    "B": ("A", "G"),
    "C": ("G", "D"),
    "D": ("C", "E"),
    "E": ("D", "G"),
    "F": ("G", "A"),
    "G": ("B", "F"),
}

SEGMENT_ORDER = [
    "A1", "B1", "C1", "D1", "E1", "F1", "G1",
    "A2", "B2", "C2", "D2", "E2", "F2", "G2"
]
# YOLOv5
sys.path.append(YOLOV5_DIR)
from models.common import DetectMultiBackend
from utils.torch_utils import select_device, smart_inference_mode
from utils.general import check_img_size, non_max_suppression
from utils.augmentations import letterbox
from utils.segment.general import process_mask_native
# PAGALBINIAI DUOMENYS
@dataclass
class ControllerState:
    segment_name: str | None = None
    led1: int | None = None
    led2: int | None = None
    raw_state_id: int | None = None
    raw_led_mode: int | None = None
    last_cmd: str = "-"
    online: bool = False
# IO
def load_layout(path):
    with open(path, "r", encoding="utf-8") as f:
        return json.load(f)
def preprocess_bgr(im0, imgsz, stride, pt):
    im = letterbox(im0, imgsz, stride=stride, auto=pt)[0]
    im = im.transpose((2, 0, 1))[::-1]
    im = np.ascontiguousarray(im)
    im = torch.from_numpy(im).float() / 255.0
    if im.ndim == 3:
        im = im.unsqueeze(0)
    return im
# DETEKCIJA
def pick_best_instance(det, masks, class_id):
    best_idx = None
    best_conf = -1.0
    for i in range(len(det)):
```

```
        cls = int(det[i, 5].item())
        conf = float(det[i, 4].item())
        if cls == class_id and conf > best_conf:
            best_conf = conf
            best_idx = i
    if best_idx is None:
        return None, None, None
    return det[best_idx], masks[best_idx].cpu().numpy().astype(bool), best_conf
def largest_contour(mask_bool):
    mask_u8 = (mask_bool.astype(np.uint8) * 255)
    contours, _ = cv2.findContours(mask_u8, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    if not contours:
        return None
    return max(contours, key=cv2.contourArea)
# GEOMETRIJA
def clip_polygon_to_image(poly, w, h):
    poly = poly.copy().astype(np.float32)
    poly[:, 0] = np.clip(poly[:, 0], 0, w - 1)
    poly[:, 1] = np.clip(poly[:, 1], 0, h - 1)
    return poly
def polygon_mask(shape_hw, poly):
    h, w = shape_hw
    mask = np.zeros((h, w), dtype=np.uint8)
    pts = np.round(poly).astype(np.int32)
    cv2.fillPoly(mask, [pts], 255)
    return mask
def polygon_inside_mask_ratio(poly, mask_bool):
    h, w = mask_bool.shape[:2]
    pm = polygon_mask((h, w), clip_polygon_to_image(poly, w, h))
    total = np.count_nonzero(pm)
    if total == 0:
        return 0.0
    inside = np.count_nonzero((pm > 0) & mask_bool)
    return inside / total
def rect_norm_to_poly(rect_norm):
    x1, y1, x2, y2 = rect_norm
    return np.array([
        [x1, y1],
        [x2, y1],
        [x2, y2],
        [x1, y2]
    ], dtype=np.float32)
def apply_homography_to_poly(poly, H):
    poly = np.asarray(poly, dtype=np.float32).reshape(-1, 1, 2)
    out = cv2.perspectiveTransform(poly, H).reshape(-1, 2)
    return out
def apply_affine_to_poly(poly, A):
    poly = np.asarray(poly, dtype=np.float32)
    ones = np.ones((poly.shape[0], 1), dtype=np.float32)
    homo = np.hstack([poly, ones])
    out = homo @ A.T
    return out.astype(np.float32)
def order_points_clockwise(pts):
    pts = np.asarray(pts, dtype=np.float32)
    c = np.mean(pts, axis=0)
    angles = np.arctan2(pts[:, 1] - c[1], pts[:, 0] - c[0])
    idx = np.argsort(angles)
    pts = pts[idx]
    start = np.argmin(pts[:, 0] + pts[:, 1])
    pts = np.roll(pts, -start, axis=0)
    return pts.astype(np.float32)
def all_quad_orderings(quad):
    quad = order_points_clockwise(quad)
    outs = []
    for k in range(4):
        outs.append(np.roll(quad, -k, axis=0))
    quad_rev = np.array([quad[0], quad[3], quad[2], quad[1]], dtype=np.float32)
    quad_rev = order_points_clockwise(quad_rev)
```

```
    for k in range(4):
        outs.append(np.roll(quad_rev, -k, axis=0))
    uniq = []
    for q in outs:
        is_new = True
        for u in uniq:
            if np.allclose(q, u, atol=1e-3):
                is_new = False
                break
        if is_new:
            uniq.append(q)
    return uniq
def approx_quad_from_contour(cnt):
    peri = cv2.arcLength(cnt, True)
    for frac in [0.01, 0.015, 0.02, 0.025, 0.03, 0.04, 0.05]:
        approx = cv2.approxPolyDP(cnt, frac * peri, True)
        if len(approx) == 4:
            return order_points_clockwise(approx[:, 0, :].astype(np.float32))
    return None
def pcb_quad_from_mask(mask_bool):
    cnt = largest_contour(mask_bool)
    if cnt is None:
        return None
    quad = approx_quad_from_contour(cnt)
    if quad is not None:
        return quad
    rect = cv2.minAreaRect(cnt)
    box = cv2.boxPoints(rect).astype(np.float32)
    return order_points_clockwise(box)
def seg_quad_from_mask(mask_bool):
    cnt = largest_contour(mask_bool)
    if cnt is None:
        return None
    rect = cv2.minAreaRect(cnt)
    box = cv2.boxPoints(rect).astype(np.float32)
    return order_points_clockwise(box)
def canonical_board_quad():
    return np.array([
        [0.0, 0.0],
        [1.0, 0.0],
        [1.0, 1.0],
        [0.0, 1.0]
    ], dtype=np.float32)
def canonical_screen_quad(layout):
    return rect_norm_to_poly(layout["SCREEN"]).astype(np.float32)
def poly_center(poly):
    return np.mean(np.asarray(poly, dtype=np.float32), axis=0)
def polygon_iou(poly1, poly2, shape_hw):
    h, w = shape_hw
    m1 = polygon_mask((h, w), poly1)
    m2 = polygon_mask((h, w), poly2)
    inter = np.count_nonzero((m1 > 0) & (m2 > 0))
    union = np.count_nonzero((m1 > 0) | (m2 > 0))
    if union == 0:
        return 0.0
    return inter / union
# ORIENTACIJA
def choose_best_pcb_homography(layout, pcb_quad_detected, seg_quad_detected, seg_mask, image_shape):
    h, w = image_shape[:2]
    board_src = canonical_board_quad()
    screen_src = canonical_screen_quad(layout)
    best = None
    best_score = -1e9
    for quad_ordered in all_quad_orderings(pcb_quad_detected):
        H = cv2.getPerspectiveTransform(board_src, quad_ordered)
        screen_proj = apply_homography_to_poly(screen_src, H)
        screen_proj = clip_polygon_to_image(screen_proj, w, h)
        iou = polygon_iou(screen_proj, seg_quad_detected, (h, w))
```

```
            inside = polygon_inside_mask_ratio(screen_proj, seg_mask)
            c1 = np.mean(screen_proj, axis=0)
            c2 = np.mean(seg_quad_detected, axis=0)
            dist = np.linalg.norm(c1 - c2)
            dist_score = -dist / max(h, w)
            score = 4.5 * iou + 2.5 * inside + dist_score
            if score > best_score:
                best_score = score
                best = {
                    "H_pcb": H,
                    "pcb_quad": quad_ordered,
                    "screen_proj": screen_proj,
                    "score": score,
                    "iou": iou,
                    "inside": inside
                }
    return best
def choose_seg_order_like_reference(seg_quad_detected, screen_proj):
    best = None
    best_err = 1e18
    for q in all_quad_orderings(seg_quad_detected):
        err = float(np.mean(np.sum((q - screen_proj) ** 2, axis=1)))
        if err < best_err:
            best_err = err
            best = q.copy()
    return best
# ROI TRANSFORMACIJOS
def transform_all_rois_by_pcb(layout, H_pcb, image_shape):
    h, w = image_shape[:2]
    out = {}
    for name, rect in layout.items():
        poly_ref = rect_norm_to_poly(rect)
        poly_cur = apply_homography_to_poly(poly_ref, H_pcb)
        poly_cur = clip_polygon_to_image(poly_cur, w, h)
        out[name] = poly_cur
    return out
def refine_rois_by_screen(rois_from_pcb, seg_quad_detected, image_shape):
    h, w = image_shape[:2]
    src_screen = rois_from_pcb["SCREEN"].astype(np.float32)
    dst_screen = choose_seg_order_like_reference(seg_quad_detected, src_screen).astype(np.float32)
    H_refine = cv2.getPerspectiveTransform(src_screen, dst_screen)
    out = {}
    for name, poly_src in rois_from_pcb.items():
        poly_dst = apply_homography_to_poly(poly_src, H_refine)
        poly_dst = clip_polygon_to_image(poly_dst, w, h)
        out[name] = poly_dst
    return out, dst_screen, H_refine
def estimate_pcb_quad_from_good_rois(layout, transformed_rois, image_shape):
    h, w = image_shape[:2]
    src = np.array([
        poly_center(rect_norm_to_poly(layout["SCREEN"])),
        poly_center(rect_norm_to_poly(layout["LED_1"])),
        poly_center(rect_norm_to_poly(layout["LED_2"])),
    ], dtype=np.float32)
    dst = np.array([
        poly_center(transformed_rois["SCREEN"]),
        poly_center(transformed_rois["LED_1"]),
        poly_center(transformed_rois["LED_2"]),
    ], dtype=np.float32)
    A = cv2.getAffineTransform(src, dst)
    pcb_ref = canonical_board_quad()
    pcb_draw = apply_affine_to_poly(pcb_ref, A)
    pcb_draw = clip_polygon_to_image(pcb_draw, w, h)
    pcb_draw = order_points_clockwise(pcb_draw)
    return pcb_draw
# ROI BRANDUOLIAI
def central_subrect_poly(poly, xr1, yr1, xr2, yr2):
    tl, tr, br, bl = poly.astype(np.float32)
```

```
    def interp(u, v):
        top = tl + u * (tr - tl)
        bot = bl + u * (br - bl)
        return top + v * (bot - top)
    return np.array([
        interp(xr1, yr1),
        interp(xr2, yr1),
        interp(xr2, yr2),
        interp(xr1, yr2),
    ], dtype=np.float32)
def is_vertical_segment(name):
    return name[0] in {"B", "C", "E", "F"}
def segment_core_poly(name, poly):
    if is_vertical_segment(name):
        return central_subrect_poly(poly, VSEG_X1, VSEG_Y1, VSEG_X2, VSEG_Y2)
    return central_subrect_poly(poly, HSEG_X1, HSEG_Y1, HSEG_X2, HSEG_Y2)
# SPALVINĖ ANALIZĖ
def color_stats_in_polygon(img_bgr, poly):
    h, w = img_bgr.shape[:2]
    poly = clip_polygon_to_image(poly, w, h)
    mask = polygon_mask((h, w), poly)
    if cv2.countNonZero(mask) == 0:
        return {
            "mean_h": 0.0,
            "mean_s": 0.0,
            "mean_v": 0.0,
            "mean_r": 0.0,
            "mean_g": 0.0,
            "mean_b": 0.0,
            "red_excess": 0.0,
            "blue_excess": 0.0,
            "rgb_spread": 0.0,
            "whiteness": 0.0,
            "brightness_pct": 0.0,
        }
    pix = mask > 0
    hsv = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2HSV)
    H, S, V = cv2.split(hsv)
    B, G, R = cv2.split(img_bgr)
    mean_h = float(np.mean(H[pix]))
    mean_s = float(np.mean(S[pix]))
    mean_v = float(np.mean(V[pix]))
    mean_r = float(np.mean(R[pix]))
    mean_g = float(np.mean(G[pix]))
    mean_b = float(np.mean(B[pix]))
    red_excess = mean_r - 0.5 * (mean_g + mean_b)
    blue_excess = mean_b - 0.5 * (mean_g + mean_r)
    rgb_spread = (abs(mean_r - mean_g) + abs(mean_r - mean_b) + abs(mean_g - mean_b)) / 3.0
    whiteness = mean_v - 0.5 * rgb_spread
    brightness_pct = np.clip((mean_v / 255.0) * 100.0, 0.0, 100.0)
    return {
        "mean_h": mean_h,
        "mean_s": mean_s,
        "mean_v": mean_v,
        "mean_r": mean_r,
        "mean_g": mean_g,
        "mean_b": mean_b,
        "red_excess": red_excess,
        "blue_excess": blue_excess,
        "rgb_spread": rgb_spread,
        "whiteness": whiteness,
        "brightness_pct": float(brightness_pct),
    }
def color_stats_top_bright_in_polygon(img_bgr, poly, top_frac=LED_TOP_BRIGHT_FRAC, min_pixels=LED_MIN_PIXELS):
    h, w = img_bgr.shape[:2]
    poly = clip_polygon_to_image(poly, w, h)
    mask = polygon_mask((h, w), poly)
    if cv2.countNonZero(mask) == 0:
```

```
        return color_stats_in_polygon(img_bgr, poly)
    hsv = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2HSV)
    H, S, V = cv2.split(hsv)
    B, G, R = cv2.split(img_bgr)
    pix = mask > 0
    vvals = V[pix]
    if vvals.size == 0:
        return color_stats_in_polygon(img_bgr, poly)
    k = max(int(vvals.size * top_frac), min_pixels)
    k = min(k, vvals.size)
    if k < vvals.size:
        thr = np.partition(vvals, vvals.size - k)[vvals.size - k]
        top_mask = pix & (V >= thr)
    else:
        top_mask = pix
    if np.count_nonzero(top_mask) < min_pixels:
        top_mask = pix
    mean_h = float(np.mean(H[top_mask]))
    mean_s = float(np.mean(S[top_mask]))
    mean_v = float(np.mean(V[top_mask]))
    mean_r = float(np.mean(R[top_mask]))
    mean_g = float(np.mean(G[top_mask]))
    mean_b = float(np.mean(B[top_mask]))
    red_excess = mean_r - 0.5 * (mean_g + mean_b)
    blue_excess = mean_b - 0.5 * (mean_g + mean_r)
    rgb_spread = (abs(mean_r - mean_g) + abs(mean_r - mean_b) + abs(mean_g - mean_b)) / 3.0
    whiteness = mean_v - 0.5 * rgb_spread
    brightness_pct = np.clip((mean_v / 255.0) * 100.0, 0.0, 100.0)
    return {
        "mean_h": mean_h,
        "mean_s": mean_s,
        "mean_v": mean_v,
        "mean_r": mean_r,
        "mean_g": mean_g,
        "mean_b": mean_b,
        "red_excess": red_excess,
        "blue_excess": blue_excess,
        "rgb_spread": rgb_spread,
        "whiteness": whiteness,
        "brightness_pct": float(brightness_pct),
    }
def compute_segment_thresholds(img_bgr, transformed_rois):
    white_scores = []
    vals = []
    spreads = []
    for name in transformed_rois.keys():
        if name not in SEGMENT_NAMES or name == "SCREEN":
            continue
        core = segment_core_poly(name, transformed_rois[name])
        st = color_stats_in_polygon(img_bgr, core)
        white_score = 0.80 * st["whiteness"] + 0.20 * st["mean_v"]
        white_scores.append(white_score)
        vals.append(st["mean_v"])
        spreads.append(st["rgb_spread"])
    if not white_scores:
        return SEG_WHITE_MIN, SEG_V_MIN, SEG_SPREAD_MAX
    white_scores = np.array(white_scores, dtype=np.float32)
    vals = np.array(vals, dtype=np.float32)
    spreads = np.array(spreads, dtype=np.float32)
    white_thr = float(np.percentile(white_scores, 25) + 0.10 * (np.percentile(white_scores, 70) - np.percentile(white_scores, 25)))
    v_thr = float(np.percentile(vals, 25) + 0.10 * (np.percentile(vals, 70) - np.percentile(vals, 25)))
    spread_thr = float(np.percentile(spreads, 80) + 0.05 * (np.percentile(spreads, 92) - np.percentile(spreads, 80)))
    white_thr = max(SEG_WHITE_MIN, min(120.0, white_thr))
    v_thr = max(SEG_V_MIN, min(140.0, v_thr))
    spread_thr = max(12.0, min(SEG_SPREAD_MAX, spread_thr))
    return white_thr, v_thr, spread_thr
def classify_segment(img_bgr, name, poly, white_thr, v_thr, spread_thr):
    core = segment_core_poly(name, poly)
```

```
    st = color_stats_in_polygon(img_bgr, core)
    white_score = 0.80 * st["whiteness"] + 0.20 * st["mean_v"]
    on = 1 if (
        white_score >= white_thr and
        st["mean_v"] >= v_thr and
        st["rgb_spread"] <= spread_thr
    ) else 0
    return {
        "state": on,
        "core_poly": core,
        "white_score": round(float(white_score), 2),
        "mean_h": round(float(st["mean_h"]), 2),
        "mean_s": round(float(st["mean_s"]), 2),
        "mean_v": round(float(st["mean_v"]), 2),
        "rgb_spread": round(float(st["rgb_spread"]), 2),
        "whiteness": round(float(st["whiteness"]), 2),
        "brightness_pct": round(float(st["brightness_pct"]), 1),
    }
def classify_led_red(img_bgr, poly):
    st = color_stats_top_bright_in_polygon(img_bgr, poly)
    mean_r = float(st["mean_r"])
    mean_g = float(st["mean_g"])
    mean_b = float(st["mean_b"])
    red_purity = mean_r / (mean_r + mean_g + mean_b + 1e-6)
    red_ratio_g = mean_r / (mean_g + 1e-6)
    red_ratio_b = mean_r / (mean_b + 1e-6)
    on = 1 if (
        mean_r >= 70.0 and
        st["mean_v"] >= 50.0 and
        st["mean_s"] >= 80.0 and
        red_purity >= 0.52 and
        red_ratio_g >= 1.8 and
        red_ratio_b >= 1.8
    ) else 0
    return {
        "state": on,
        "score": round(float(0.60 * mean_r + 0.25 * st["mean_s"] + 0.15 * st["mean_v"]), 2),
        "mean_h": round(float(st["mean_h"]), 2),
        "mean_s": round(float(st["mean_s"]), 2),
        "mean_v": round(float(st["mean_v"]), 2),
        "mean_r": round(float(mean_r), 2),
        "mean_g": round(float(mean_g), 2),
        "mean_b": round(float(mean_b), 2),
        "red_excess": round(float(st["red_excess"]), 2),
        "blue_excess": round(float(st["blue_excess"]), 2),
        "red_purity": round(float(red_purity), 3),
        "red_ratio_g": round(float(red_ratio_g), 2),
        "red_ratio_b": round(float(red_ratio_b), 2),
        "brightness_pct": round(float(st["brightness_pct"]), 1),
    }
def classify_led_blue(img_bgr, poly):
    st = color_stats_top_bright_in_polygon(img_bgr, poly)
    mean_r = float(st["mean_r"])
    mean_g = float(st["mean_g"])
    mean_b = float(st["mean_b"])
    blue_purity = mean_b / (mean_r + mean_g + mean_b + 1e-6)
    blue_ratio_r = mean_b / (mean_r + 1e-6)
    blue_ratio_g = mean_b / (mean_g + 1e-6)
    on = 1 if (
        mean_b >= 70.0 and
        st["mean_v"] >= 50.0 and
        st["mean_s"] >= 80.0 and
        blue_purity >= 0.52 and
        blue_ratio_r >= 1.8 and
        blue_ratio_g >= 1.8
    ) else 0
    return {
        "state": on,
```

```
            "score": round(float(0.60 * mean_b + 0.25 * st["mean_s"] + 0.15 * st["mean_v"]), 2),
            "mean_h": round(float(st["mean_h"]), 2),
            "mean_s": round(float(st["mean_s"]), 2),
            "mean_v": round(float(st["mean_v"]), 2),
            "mean_r": round(float(mean_r), 2),
            "mean_g": round(float(mean_g), 2),
            "mean_b": round(float(mean_b), 2),
            "red_excess": round(float(st["red_excess"]), 2),
            "blue_excess": round(float(st["blue_excess"]), 2),
            "blue_purity": round(float(blue_purity), 3),
            "blue_ratio_r": round(float(blue_ratio_r), 2),
            "blue_ratio_g": round(float(blue_ratio_g), 2),
            "brightness_pct": round(float(st["brightness_pct"]), 1),
        }
    # POST-PROCESSING
    def suppress_to_single_segment(states, suffix):
        raw_on = {seg: int(states[f"{seg}{suffix}"]["state"]) for seg in DIGIT_SEGMENTS}
        raw_score = {seg: float(states[f"{seg}{suffix}"]["white_score"]) for seg in DIGIT_SEGMENTS}
        on_count = sum(raw_on.values())
        if on_count <= 1:
            return states
        support = {}
        for center in DIGIT_SEGMENTS:
            n1, n2 = CENTER_NEIGHBORS[center]
            s = (
                1.00 * raw_score[center] +
                0.35 * raw_score[n1] +
                0.35 * raw_score[n2]
            )
            if raw_on[center]:
                s += 12.0
            if raw_on[center] and raw_on[n1] and raw_on[n2]:
                s += 18.0
            support[center] = s
        winner = max(support, key=support.get)
        for seg in DIGIT_SEGMENTS:
            states[f"{seg}{suffix}"]["state_raw"] = states[f"{seg}{suffix}"]["state"]
            states[f"{seg}{suffix}"]["support_score"] = round(float(support[seg]), 2)
            states[f"{seg}{suffix}"]["state"] = 1 if seg == winner else 0
        return states
    def extract_predicted_segment(states):
        active = []
        for seg in SEGMENT_ORDER:
            if int(states[seg]["state"]) == 1:
                active.append(seg)
        if len(active) == 1:
            return active[0]
        if len(active) == 0:
            return ""
        return "|".join(active)
    def get_segment_brightness(states, segment_name):
        if not segment_name:
            return 0.0
        if "|" in segment_name:
            parts = [p for p in segment_name.split("|") if p in states]
            if not parts:
                return 0.0
            return round(float(np.mean([states[p]["brightness_pct"] for p in parts])), 1)
        if segment_name in states:
            return float(states[segment_name]["brightness_pct"])
        return 0.0
    #FILTRAS
    class StableValue:
        def __init__(self, initial_value=None, confirm_frames=3):
            self.stable = initial_value
            self.candidate = initial_value
            self.candidate_count = 0
            self.confirm_frames = confirm_frames
```

```
    def update(self, new_value):
        if self.stable is None:
            self.stable = new_value
            self.candidate = new_value
            self.candidate_count = 0
            return self.stable
        if new_value == self.stable:
            self.candidate = new_value
            self.candidate_count = 0
            return self.stable
        if new_value != self.candidate:
            self.candidate = new_value
            self.candidate_count = 1
            return self.stable
        self.candidate_count += 1
        if self.candidate_count >= self.confirm_frames:
            self.stable = self.candidate
            self.candidate_count = 0
        return self.stable
class TemporalStateFilter:
    def __init__(self, confirm_frames=3):
        self.segment = StableValue("", confirm_frames)
        self.segment_brightness = StableValue(0.0, confirm_frames)
        self.led1 = StableValue(0, confirm_frames)
        self.led1_brightness = StableValue(0.0, confirm_frames)
        self.led2 = StableValue(0, confirm_frames)
        self.led2_brightness = StableValue(0.0, confirm_frames)
    def update(self, states):
        segment_now = extract_predicted_segment(states)
        seg_bri_now = get_segment_brightness(states, segment_now)
        led1_now = int(states["LED_1"]["state"])
        led1_bri_now = float(states["LED_1"]["brightness_pct"])
        led2_now = int(states["LED_2"]["state"])
        led2_bri_now = float(states["LED_2"]["brightness_pct"])
        stable_segment = self.segment.update(segment_now)
        stable_seg_bri = self.segment_brightness.update(seg_bri_now)
        stable_led1 = self.led1.update(led1_now)
        stable_led1_bri = self.led1_brightness.update(led1_bri_now)
        stable_led2 = self.led2.update(led2_now)
        stable_led2_bri = self.led2_brightness.update(led2_bri_now)
        return {
            "segment": stable_segment,
            "segment_brightness": stable_seg_bri,
            "led1": stable_led1,
            "led1_brightness": stable_led1_bri,
            "led2": stable_led2,
            "led2_brightness": stable_led2_bri,
        }
# JUDESIO / STATIŠKUMO KONTROLĖ
class FreezeOnStillness:
    def __init__(self, static_threshold, static_frames_to_lock, unlock_threshold, blur_ksize=(9, 9)):
        self.static_threshold = float(static_threshold)
        self.static_frames_to_lock = int(static_frames_to_lock)
        self.unlock_threshold = float(unlock_threshold)
        self.blur_ksize = tuple(blur_ksize)
        self.prev_gray = None
        self.static_count = 0
        self.locked = False
        self.last_diff_value = 999.0
    def _prepare(self, frame_bgr):
        gray = cv2.cvtColor(frame_bgr, cv2.COLOR_BGR2GRAY)
        gray = cv2.GaussianBlur(gray, self.blur_ksize, 0)
        return gray
    def update(self, frame_bgr):
        gray = self._prepare(frame_bgr)
        if self.prev_gray is None:
            self.prev_gray = gray
            self.static_count = 0
```

```
            self.locked = False
            self.last_diff_value = 999.0
            return {
                "locked": self.locked,
                "diff_value": self.last_diff_value,
                "just_locked": False,
                "just_unlocked": False,
            }
        diff = cv2.absdiff(gray, self.prev_gray)
        diff_value = float(np.mean(diff))
        self.last_diff_value = diff_value
        just_locked = False
        just_unlocked = False
        if self.locked:
            if diff_value >= self.unlock_threshold:
                self.locked = False
                self.static_count = 0
                just_unlocked = True
        else:
            if diff_value <= self.static_threshold:
                self.static_count += 1
                if self.static_count >= self.static_frames_to_lock:
                    self.locked = True
                    just_locked = True
            else:
                self.static_count = 0
        self.prev_gray = gray
        return {
            "locked": self.locked,
            "diff_value": diff_value,
            "just_locked": just_locked,
            "just_unlocked": just_unlocked,
        }
# VALDIKLIO SIGNALO PARSINIMAS
def controller_segment_from_state_id(state_id):
    if state_id is None:
        return None
    if 100 <= state_id <= 113:
        k = state_id - 100
        digit_idx = k // 7
        seg_idx = k % 7
        suffix = "1" if digit_idx == 0 else "2"
        seg_letter = DIGIT_SEGMENTS[seg_idx]
        return f"{seg_letter}{suffix}"
    return None
def controller_leds_from_mode(mode):
    if mode is None:
        return None, None
    mode &= 3
    led1 = 1 if mode in (1, 2) else 0
    led2 = 1 if mode in (2, 3) else 0
    return led1, led2
def update_controller_state_from_line(ctrl_state, line):
    line = line.strip()
    if not line:
        return ctrl_state
    ctrl_state.last_cmd = line
    ctrl_state.online = True
    if line.startswith("SETSTATE"):
        try:
            state_id = int(line[8:].strip())
            state_id = max(0, min(113, state_id))
            ctrl_state.raw_state_id = state_id
            ctrl_state.segment_name = controller_segment_from_state_id(state_id)
        except Exception:
            pass
    elif line.startswith("SETVAL"):
        try:
```

```
            value = int(line[6:].strip())
            value = max(0, min(99, value))
            ctrl_state.raw_state_id = value
            ctrl_state.segment_name = None
        except Exception:
            pass
    elif line.startswith("SETLED"):
        try:
            led_mode = int(line[6:].strip()) & 3
            ctrl_state.raw_led_mode = led_mode
            ctrl_state.led1, ctrl_state.led2 = controller_leds_from_mode(led_mode)
        except Exception:
            pass
    return ctrl_state
def read_controller_serial(ser, ctrl_state):
    if ser is None:
        return ctrl_state
    try:
        while ser.in_waiting > 0:
            raw = ser.readline().decode("utf-8", errors="ignore")
            ctrl_state = update_controller_state_from_line(ctrl_state, raw)
    except Exception:
        ctrl_state.online = False
    return ctrl_state
def open_serial_if_possible():
    if SERIAL_PORT is None or serial is None:
        return None
    try:
        ser = serial.Serial(
            SERIAL_PORT,
            SERIAL_BAUD,
            timeout=SERIAL_TIMEOUT,
            write_timeout=0,
            dsrdtr=False,
            rtscts=False,
        )
        try:
            ser.setDTR(False)
            ser.setRTS(False)
        except Exception:
            pass
        time.sleep(SERIAL_OPEN_WAIT_S)
        try:
            ser.reset_input_buffer()
        except Exception:
            pass
        return ser
    except Exception:
        return None
def calc_match_percent(stable_values, ctrl_state):
    checks = []
    if ctrl_state.segment_name is not None:
        checks.append(1 if stable_values["segment"] == ctrl_state.segment_name else 0)
    if ctrl_state.led1 is not None:
        checks.append(1 if int(stable_values["led1"]) == int(ctrl_state.led1) else 0)
    if ctrl_state.led2 is not None:
        checks.append(1 if int(stable_values["led2"]) == int(ctrl_state.led2) else 0)
    if not checks:
        return None
    return round(100.0 * sum(checks) / len(checks), 1)
# OPTINIS SEKIMAS
def build_tracking_mask(frame_shape, rois):
    h, w = frame_shape[:2]
    mask = np.zeros((h, w), dtype=np.uint8)
    keys = ["SCREEN", "LED_1", "LED_2"]
    for key in keys:
        poly = rois.get(key)
        if poly is not None:
```

```
            pts = np.round(poly).astype(np.int32)
            cv2.fillPoly(mask, [pts], 255)
    return mask
def detect_tracking_points(gray, rois):
    mask = build_tracking_mask(gray.shape, rois)
    pts = cv2.goodFeaturesToTrack(
        gray,
        maxCorners=FLOW_MAX_CORNERS,
        qualityLevel=FLOW_QUALITY_LEVEL,
        minDistance=FLOW_MIN_DISTANCE,
        mask=mask,
        blockSize=FLOW_BLOCK_SIZE,
    )
    return pts
def track_rois_optical_flow(prev_frame_bgr, frame_bgr, rois):
    prev_gray = cv2.cvtColor(prev_frame_bgr, cv2.COLOR_BGR2GRAY)
    gray = cv2.cvtColor(frame_bgr, cv2.COLOR_BGR2GRAY)
    prev_pts = detect_tracking_points(prev_gray, rois)
    if prev_pts is None or len(prev_pts) < FLOW_MIN_GOOD_POINTS:
        return None
    next_pts, status, err = cv2.calcOpticalFlowPyrLK(
        prev_gray, gray, prev_pts, None,
        winSize=FLOW_WIN_SIZE,
        maxLevel=FLOW_MAX_LEVEL,
        criteria=(cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT, 25, 0.01),
    )
    if next_pts is None or status is None:
        return None
    status = status.reshape(-1)
    err = err.reshape(-1) if err is not None else np.full(len(status), 9999.0, dtype=np.float32)
    good_prev = prev_pts[status == 1].reshape(-1, 2)
    good_next = next_pts[status == 1].reshape(-1, 2)
    good_err = err[status == 1]
    if len(good_prev) < FLOW_MIN_GOOD_POINTS:
        return None
    good_mask = good_err <= FLOW_MAX_ERR
    good_prev = good_prev[good_mask]
    good_next = good_next[good_mask]
    if len(good_prev) < FLOW_MIN_GOOD_POINTS:
        return None
    A, inliers = cv2.estimateAffinePartial2D(
        good_prev,
        good_next,
        method=cv2.RANSAC,
        ransacReprojThreshold=FLOW_INLIER_RANSAC,
        maxIters=2000,
        confidence=0.99,
        refineIters=10,
    )
    if A is None:
        return None
    inlier_ratio = 0.0
    if inliers is not None and len(inliers) > 0:
        inlier_ratio = float(np.mean(inliers.astype(np.float32)))
    h, w = frame_bgr.shape[:2]
    tracked = {}
    for name, poly in rois.items():
        new_poly = apply_affine_to_poly(poly, A)
        new_poly = clip_polygon_to_image(new_poly, w, h)
        tracked[name] = new_poly
    pcb_draw_quad = None
    if "SCREEN" in tracked and "LED_1" in tracked and "LED_2" in tracked:
        tracked_for_pcb = {
            "SCREEN": tracked["SCREEN"],
            "LED_1": tracked["LED_1"],
            "LED_2": tracked["LED_2"],
        }
        pcb_draw_quad = estimate_pcb_quad_from_good_rois(G_LAYOUT, tracked_for_pcb, frame_bgr.shape)
```

```
    return {
        "rois": tracked,
        "affine": A,
        "points_total": int(len(prev_pts)),
        "points_good": int(len(good_prev)),
        "inlier_ratio": inlier_ratio,
        "pcb_draw_quad": pcb_draw_quad,
    }
# VIENO KADRO ATPAŽINIMAS
def classify_from_existing_rois(im0, transformed_rois):
    seg_white_thr, seg_v_thr, seg_spread_thr = compute_segment_thresholds(im0, transformed_rois)
    states = {}
    for name in SEGMENT_ORDER:
        states[name] = classify_segment(
            im0, name, transformed_rois[name],
            seg_white_thr, seg_v_thr, seg_spread_thr
        )
    states = suppress_to_single_segment(states, suffix="1")
    states = suppress_to_single_segment(states, suffix="2")
    states["LED_1"] = classify_led_red(im0, transformed_rois["LED_1"])
    states["LED_2"] = classify_led_blue(im0, transformed_rois["LED_2"])
    return states, seg_white_thr, seg_v_thr, seg_spread_thr
def full_detect_and_build_rois(im0, model, layout, imgsz, stride, pt, device):
    im = preprocess_bgr(im0, imgsz, stride, pt).to(device)
    out = model(im, augment=False, visualize=False)
    if isinstance(out, (list, tuple)):
        pred = out[0]
        proto = out[1]
    else:
        return None
    pred = non_max_suppression(
        pred,
        CONF_THRES,
        IOU_THRES,
        classes=None,
        agnostic=False,
        max_det=100,
        nm=32,
    )
    det = pred[0]
    if det is None or len(det) == 0:
        return None
    masks = process_mask_native(proto[0], det[:, 6:], det[:, :4], im0.shape[:2])
    _, pcb_mask, pcb_conf = pick_best_instance(det, masks, PCB_CLASS_ID)
    _, seg_mask, seg_conf = pick_best_instance(det, masks, SEG_CLASS_ID)
    if pcb_mask is None or seg_mask is None:
        return None
    pcb_quad = pcb_quad_from_mask(pcb_mask)
    seg_quad = seg_quad_from_mask(seg_mask)
    if pcb_quad is None or seg_quad is None:
        return None
    best_fit = choose_best_pcb_homography(
        layout=layout,
        pcb_quad_detected=pcb_quad,
        seg_quad_detected=seg_quad,
        seg_mask=seg_mask,
        image_shape=im0.shape
    )
    if best_fit is None:
        return None
    H_pcb = best_fit["H_pcb"]
    rois_global = transform_all_rois_by_pcb(layout, H_pcb, im0.shape)
    transformed_rois, _, _ = refine_rois_by_screen(rois_global, seg_quad, im0.shape)
    states, seg_white_thr, seg_v_thr, seg_spread_thr = classify_from_existing_rois(im0, transformed_rois)
    pcb_draw_quad = estimate_pcb_quad_from_good_rois(layout, transformed_rois, im0.shape)
    return {
        "states": states,
        "transformed_rois": transformed_rois,
```

```
        "pcb_draw_quad": pcb_draw_quad,
        "pcb_conf": float(pcb_conf),
        "seg_conf": float(seg_conf),
        "fit_score": float(best_fit["score"]),
        "screen_iou": float(best_fit["iou"]),
        "seg_white_thr": float(seg_white_thr),
        "seg_v_thr": float(seg_v_thr),
        "seg_spread_thr": float(seg_spread_thr),
        "tracker_points_good": 0,
        "tracker_inlier_ratio": 0.0,
    }
def update_geometry_by_tracking(prev_frame, frame, cache):
    tracked = track_rois_optical_flow(prev_frame, frame, cache["transformed_rois"])
    if tracked is None:
        return None
    cache2 = dict(cache)
    cache2["transformed_rois"] = tracked["rois"]
    if tracked["pcb_draw_quad"] is not None:
        cache2["pcb_draw_quad"] = tracked["pcb_draw_quad"]
    cache2["tracker_points_good"] = tracked["points_good"]
    cache2["tracker_inlier_ratio"] = tracked["inlier_ratio"]
    states, seg_white_thr, seg_v_thr, seg_spread_thr = classify_from_existing_rois(frame, cache2["transformed_rois"])
    cache2["states"] = states
    cache2["seg_white_thr"] = seg_white_thr
    cache2["seg_v_thr"] = seg_v_thr
    cache2["seg_spread_thr"] = seg_spread_thr
    return cache2
# BRAIŽYMAS
def draw_poly(img, poly, color, thickness=2):
    pts = np.round(poly).astype(np.int32)
    cv2.polylines(img, [pts], True, color, thickness, cv2.LINE_AA)
def put_text(img, text, x, y, color=(255, 255, 255), scale=0.8, thickness=2):
    cv2.putText(img, text, (x, y), cv2.FONT_HERSHEY_SIMPLEX, scale, (0, 0, 0), thickness + 2, cv2.LINE_AA)
    cv2.putText(img, text, (x, y), cv2.FONT_HERSHEY_SIMPLEX, scale, color, thickness, cv2.LINE_AA)
def draw_only_active_segment(vis, cache, stable_values):
    if cache is None:
        return
    seg_name = stable_values["segment"]
    if not seg_name or seg_name not in cache["transformed_rois"]:
        return
    seg_roi = cache["transformed_rois"][seg_name]
    draw_poly(vis, seg_roi, (0, 255, 0), 3)
    c = poly_center(seg_roi).astype(int)
    put_text(vis, seg_name, int(c[0]), int(c[1]), (0, 255, 0), scale=0.65, thickness=2)
def calc_single_item_match(detected, expected):
    if expected is None:
        return "-"
    return "100 %" if detected == expected else "0 %"
def draw_overlay(frame, cache, mode_text, stable_values, ctrl_state, fps, diff_value, freeze_locked):
    vis = frame.copy()
    if cache is not None:
        rois = cache["transformed_rois"]
        draw_poly(vis, cache["pcb_draw_quad"], (255, 200, 0), 3)
        draw_poly(vis, rois["SCREEN"], (0, 255, 255), 3)
        draw_only_active_segment(vis, cache, stable_values)
        led1_color = (0, 255, 0) if stable_values["led1"] == 1 else (80, 80, 80)
        led2_color = (0, 255, 0) if stable_values["led2"] == 1 else (80, 80, 80)
        draw_poly(vis, rois["LED_1"], led1_color, 3)
        draw_poly(vis, rois["LED_2"], led2_color, 3)
        c1 = poly_center(rois["LED_1"]).astype(int)
        c2 = poly_center(rois["LED_2"]).astype(int)
        put_text(vis, "LED 1", int(c1[0]), int(c1[1]), led1_color, scale=0.7, thickness=2)
        put_text(vis, "LED 2", int(c2[0]), int(c2[1]), led2_color, scale=0.7, thickness=2)
        ctrl_seg = ctrl_state.segment_name if ctrl_state.segment_name is not None else "-"
        ctrl_led1 = ctrl_state.led1 if ctrl_state.led1 is not None else "-"
        ctrl_led2 = ctrl_state.led2 if ctrl_state.led2 is not None else "-"
        seg_match = calc_single_item_match(stable_values["segment"], ctrl_state.segment_name)
        led1_match = calc_single_item_match(int(stable_values["led1"]), ctrl_state.led1)
```

```
        led2_match = calc_single_item_match(int(stable_values["led2"]), ctrl_state.led2)
        put_text(vis, f"Seg: {stable_values['segment'] if stable_values['segment'] else '-'}", 20, 20)
        put_text(vis, f"Seg rysk: {stable_values['segment_brightness']:.1f} % | sutap su vald: {seg_match} | vald: {ctrl_seg}", 20, 50)
        put_text(vis, f"LED 1: {stable_values['led1']} | rysk: {stable_values['led1_brightness']:.1f} % | sutap: {led1_match} | vald: {ctrl_led1}", 20, 80)
        put_text(vis, f"LED 2: {stable_values['led2']} | rysk: {stable_values['led2_brightness']:.1f} % | sutap: {led2_match} | vald: {ctrl_led2}", 20, 110)
    else:
        put_text(vis, "Neaptikta", 20, 30)
    uart_txt = "ON" if ctrl_state.online else "OFF"
    port_txt = SERIAL_PORT if SERIAL_PORT is not None else "None"
    put_text(vis, f"UART: {uart_txt} | PORT: {port_txt}", 20, 145)
    put_text(vis, f"RAW state_id: {ctrl_state.raw_state_id} | led_mode: {ctrl_state.raw_led_mode}", 20, 175)
    put_text(vis, f"LAST: {ctrl_state.last_cmd}", 20, 205)
    put_text(vis, "Q - iseiti", 20, 235)
    return vis
# GLOBALUS LAYOUT TRACKINGUI
G_LAYOUT = None
# MAIN
@smart_inference_mode()
def main():
    global G_LAYOUT
    if not os.path.exists(MODEL_PATH):
        print("Nerastas modelis:", MODEL_PATH)
        return
    if not os.path.exists(ROI_LAYOUT_PATH):
        print("Nerastas roi_layout.json:", ROI_LAYOUT_PATH)
        return
    layout = load_layout(ROI_LAYOUT_PATH)
    G_LAYOUT = layout
    device = select_device("cpu")
    model = DetectMultiBackend(MODEL_PATH, device=device, dnn=False, fp16=False)
    stride, pt = model.stride, model.pt
    imgsz = check_img_size((IMG_SIZE, IMG_SIZE), s=stride)
    cam = cv2.VideoCapture(CAMERA_ID, cv2.CAP_DSHOW)
    if not cam.isOpened():
        raise RuntimeError("Kamera nepasileido")
    ser = open_serial_if_possible()
    ctrl_state = ControllerState(online=(ser is not None))
    temporal_filter = TemporalStateFilter(confirm_frames=STATE_CONFIRM_FRAMES)
    freeze_ctrl = FreezeOnStillness(
        static_threshold=STATIC_DIFF_THRESHOLD,
        static_frames_to_lock=STATIC_CONSEC_FRAMES_TO_LOCK,
        unlock_threshold=UNLOCK_MOTION_THRESHOLD,
        blur_ksize=BLUR_FOR_MOTION,
    )
    cache = None
    prev_frame = None
    frame_idx = 0
    print("Spausk Q, kad iseitum.")
    prev_t = time.perf_counter()
    while True:
        ret, frame = cam.read()
        if not ret:
            print("Kadras negautas")
            break
        now_t = time.perf_counter()
        dt = max(now_t - prev_t, 1e-6)
        fps = 1.0 / dt
        prev_t = now_t
        ctrl_state = read_controller_serial(ser, ctrl_state)
        freeze_info = freeze_ctrl.update(frame)
        freeze_locked = freeze_info["locked"]
        diff_value = freeze_info["diff_value"]
        mode_text = "Optinis sekimas"
        if cache is None:
            result = full_detect_and_build_rois(frame, model, layout, imgsz, stride, pt, device)
            if result is not None:
                cache = result
                mode_text = "Fiksavimas"
```

```
        else:
            need_full_detect = False
            if freeze_info["just_unlocked"]:
                need_full_detect = True
            elif frame_idx % DETECT_EVERY_N_FRAMES == 0:
                need_full_detect = True
            elif cache["fit_score"] < MIN_FIT_SCORE or cache["screen_iou"] < MIN_SCREEN_IOU:
                need_full_detect = True
            if freeze_locked:
                mode_text = "Nejuda"
            else:
                updated = None
                if prev_frame is not None:
                    updated = update_geometry_by_tracking(prev_frame, frame, cache)
                if updated is not None:
                    cache = updated
                    mode_text = "Optinis sekimas"
                else:
                    need_full_detect = True
                if need_full_detect:
                    result = full_detect_and_build_rois(frame, model, layout, imgsz, stride, pt, device)
                    if result is not None:
                        cache = result
                        mode_text = "Fiksavimas"
                    else:
                        mode_text = "Laikymas"
        if cache is not None:
            if freeze_locked:
                stable_values = {
                    "segment": temporal_filter.segment.stable if temporal_filter.segment.stable else "",
                    "segment_brightness": temporal_filter.segment_brightness.stable if temporal_filter.segment_brightness.stable is not None else 0.0,
                    "led1": temporal_filter.led1.stable if temporal_filter.led1.stable is not None else 0,
                    "led1_brightness": temporal_filter.led1_brightness.stable if temporal_filter.led1_brightness.stable is not None else 0.0,
                    "led2": temporal_filter.led2.stable if temporal_filter.led2.stable is not None else 0,
                    "led2_brightness": temporal_filter.led2_brightness.stable if temporal_filter.led2_brightness.stable is not None else 0.0,
                }
            else:
                stable_values = temporal_filter.update(cache["states"])
        else:
            stable_values = {
                "segment": "",
                "segment_brightness": 0.0,
                "led1": 0,
                "led1_brightness": 0.0,
                "led2": 0,
                "led2_brightness": 0.0,
            }
        vis = draw_overlay(frame, cache, mode_text, stable_values, ctrl_state, fps, diff_value, freeze_locked)
        cv2.imshow("Realaus laiko segmentu ir LED atpazinimas", vis)
        prev_frame = frame.copy()
        key = cv2.waitKey(1) & 0xFF
        if key == ord("q"):
            break
        frame_idx += 1
    if ser is not None:
        try:
            ser.close()
        except Exception:
            pass
    cam.release()
    cv2.destroyAllWindows()
if __name__ == "__main__":
    main()
```