KAUNO TECHNOLOGIJOS UNIVERSITETAS
ELEKTROS IR ELEKTRONIKOS FAKULTETAS

MARTYNAS GALGINAS

SPAUSDINTINIŲ PLOKŠČIŲ (PCB) VIZUALINIŲ DEFEKTŲ INSPEKTAVIMO METODO SUKŪRIMAS IR TYRIMAS

Baigiamasis magistro projektas

Vadovas
prof.dr. Arūnas Lipnickas

KAUNAS, 2016

KAUNO TECHNOLOGIJOS UNIVERSITETAS
ELEKTROS IR VALDYMO INŽINERIJOS FAKULTETAS
AUTOMATIKOS KATEDRA

SPAUSDINTINIŲ PLOKŠČIŲ (PCB) VIZUALINIŲ DEFEKTŲ INSPEKTAVIMO METODO SUKŪRIMAS IR TYRIMAS

Baigiamasis magistro projektas

Valdymo technologijos (kodas 621H66001)

Atliko: EMV – 4/2 gr. stud.
Martynas Galginas
2016 m. birželis..... d.

Vadovas
Arūnas Lipnickas
2016 m. birželis..... d.

Recenzentas
.....
2016 m. birželis..... d.

KAUNAS, 2016

KAUNO TECHNOLOGIJOS UNIVERSITETAS

ELEKTROS IR ELEKTRONIKOS FAKULTETAS

(Fakultetas)

Martynas Galginas

(Studento vardas, pavardė)

Valdymo technologijos (621H66001)

(Studijų programos pavadinimas, kodas)

Baigiamojo projekto „ Spausdintinių plokščių (PCB) vizualinių defektų inspektavimo metodo sukūrimas ir tyrimas “

AKADEMINIO SĄŽININGUMO DEKLARACIJA

20 16 m. gegužės 23 d.

Kaunas

Patvirtinu, kad mano **Martyno Galgino** baigiamasis projektas tema „ **Spausdintinių plokščių (PCB) vizualinių defektų inspektavimo metodo sukūrimas ir tyrimas** “ yra parašytas visiškai savarankiškai, o visi pateikti duomenys ar tyrimų rezultatai yra teisingi ir gauti sąžiningai. Šiame darbe nei viena dalis nėra plagijuota nuo jokių spausdintinių ar internetinių šaltinių, visos kitų šaltinių tiesioginės ir netiesioginės citatos nurodytos literatūros nuorodose. Įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs.

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

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

Galginas M. Spausdintinių plokščių (PCB) vizualinių defektų inspektavimo metodo sukūrimas ir tyrimas. Valdymo sistemų magistro projektas. Studijų programa 621H66001. Vadovas prof. dr. Arūnas Lipnickas. Kaunas: Kauno technologijos universiteto Elektros ir valdymo inžinerijos fakultetas, 2016, 78 p.

SANTRAUKA

Šio darbo tikslas – sukurti PCB vizualinių defektų inspektavimo metodą Matlab aplinkoje, kurio pagalba būtų galima atlikti Matlab programiniame pakete siūlomų atvaizdų apdorojimo filtrų tyrimą.

Pirmojoje darbo dalyje apžvelgiamos egzistuojančios PCB defektų inspektavimo technologijos, aprašomas PCB gamybos procesas ir defektai atsirandantys gaminimo metu.

Antrojoje dalyje aprašomas PCB vizualinių defektų inspektavimo metodas.

Trečioje dalyje tiriama matlab programiniame pakete siūlomų binarinio atvaizdo apdorojimo filtrų įtaka galutiniam rezultatui, kai yra žinomi tikslūs defektai ir jų kiekis.

Paskutinėje dalyje pateikiamos darbo išvados.

Reikšminiai žodžiai: Spausdintinė plokštė, inspektavimas, defektai, vizualinis, filtravimas, pikselis, atvaizdas, binarinis.

Galginas M. Development and analysis of method for printed circuit board (PCB) visual defects inspection. Master's project in Control Systems. Study programme 621H66001. Supervisor prof. dr. Arūnas Lipnickas. Kaunas: Faculty of Electrical and control engineering, Kaunas University of Technology, 2016, 78 p.

SUMMARY

The objective of this work is to create method for printed circuit board (PCB) visual defects inspection using Matlab software which make it possible to perform development for Matlab image processing filters.

First part of this work provides an overview of existing PCB defect inspection technology, describes PCB manufacturing process and defects which appears during manufacture

The second part of the thesis focuses development of method for printed circuit board (PCB) visual defects inspection.

In the third part analyzed MATLAB programming package offered image processing filters for binary image influence to the final result when is known defects and their precise amount.

The findings of the work are in the last part of the thesis.

Keywords : Printed circuit board,inspection, defects, visual, filtering, pixel, image, binary.

Turinys

Įvadas

Žmogus, matydamas aplinką, lengvai išskiria ir atpažįsta įvairius objektus - tai savaime suprantama užduotis, tačiau tam yra naudojama žmogaus patirtis, įgyta per ilgą laiką. Mašininė rega ar objektų atpažinimas yra sudėtingas veiksmas, iki šiol neturintis vieningų ir universalių algoritmų, gerai ar bent patenkinamai veikiančių įvairiose situacijose. Norint apdoroti vaizdą, jį reikia suskaitmeninti, t.y. gauti vaizdo duomenų masyvą, kurį apdorojant būtų galima gauti norimus rezultatus.

Skaitmeninis vaizdų apdorojimas šiuo metu yra viena iš dažniausiai sutinkamų skaitmeninių signal apdorojimo sričių, naudojama pramonėje, buityje, transporte. Skaitmeniniai prietaisai, naudojantys vaizdų apdorojimą, buityje: fotoaparatai, televizoriai, vaizdo kameros; medicininėje technikoje: echoskopai, skaitmeniniai rentgenai, oftalmologai; transporte: automatinio vairavimo, automobilių statymo sistemos, numerių atpažinimas; pramonėje: automatizuotos linijos – robotai, vaizdo apsaugos sistemos, vaizdinės inspekcijos sistemos.

Šios magistro darbo temos pasirinkimą sąlygojo darbo autoriaus praktinės veiklos patirtis elektroninę įrangą gaminančioje firmoje.

Pagrindinė magistro darbe nagrinėjama **problema** – PCB defektų identifikavimas naudojant vizualinę inspekciją, pačios sistemos sukūrimas. Dažniausiai, siekiant sutaupyti, PCB gaminančios firmos naudoja kontaktinės testavimo sistemas. Naudojant vizualines PCB defektų inspektavimo sistemas reikalingi sudėtingi algoritmai, kurie randa defektus ir juos suklasifikuoja.

Darbo objektas – etaloninės ir defektuotos PCB.

Darbo tikslas - Išanalizuoti Spausdintinių plokščių (PCB) defektus. Sukurti defektų inspektavimo sistemą naudojantis matlab programiniu paketu. Ištirti matlab programiniame pakete siūlomus vaizdo apdorojimo įrankius PCB defektų aptikimui.

Darbo struktūra: įvadas, trys skyriai, išvados, literatūros sąrašas ir priedai.

Darbo uždaviniai: Išnagrinėti literatūrą apie egzistuojančias PCB defektų inspektavimo technologijas. Aprašyti PCB gamybos procesą. Pasiūlyti vizualinių defektų inspektavimo metodą. Ištirti *Matlab* programiniame pakete siūlomų binarinio atvaizdo apdorojimo filtrų įtaką kalidų atpažinimo rezultatui, kai yra žinomi tikslūs defektai ir jų kiekis.

1. APŽVALGINĖ DALIS

1.1 PCB - spausdintinė montažinė plokštė. PCB gamyba.

Spausdintinė montažinė plokštė (angl. *Printed circuit board*) arba PCB yra plokštė ant kurios montuojami elektroniniai komponentai, kuri pagaminta iš izoliacinio ir elektrai laidaus sluoksnių. Izoliacinis sluoksnis dažniausiai būna vientisas (ištisinis), storesnis nei laidininko sluoksnis, ir sudaro mechaninį pagrindą. Dalis laidaus sluoksnio yra vienokiu ar kitokiu būdu panaikinama, kieno pasekoje yra suformuojamas elektrai laidžių takelių tinklas. Dažniausiai, prie šių takelių yra lituojamos detalės.[4]

Prieš atsirandant šioms plokštėms, buvo naudojama kabančio montažo technologija. Kabantis montažas pavadintas taip, nes viskas buvo sujungiama laidais. PCB daug lengviau paleisti į masinę gamybą, nors tam reikia daugiau laiko ir sąnaudų. Laiko sąnaudos išauga nes PCB plokštės takeliai negali kirstis tarpusavyje, turi būti elektriškai izoliuoti vienas nuo kito. Tam, kad toks uždavinys palengvėtų, takeliai išvedžiojami abiejose PCB pusėse.[6]

Egzistuoja trys pagrindinės PCB konstrukcijos: *vienpusės PCB, dvipusės PCB, daugiasluoksnės PCB*. Vienpusėse plokštėse komponentai lituojami tik vienoje pusėje. Kai lituojamų komponentų fiziškai nebeįmanoma sutalpinti vienoje plokštės pusėje, naudojama dvipusė PCB. Elektrinis laidumas tarp plokštės pusių gali būti realizuojamas pragręžiant skyles takeliuose ir esamą ertmę padengiant elektriškai laidžiu sluoksniu. Daugiasluoksnės PCB yra pagamintos sujungiant *n* sluoksnių elektriškai laidžių takelių, prieš tai juos atskyrus elektrai nelaidžiais sluoksniais.[6]

Elektroniniai komponentai prie plokštės prijungiami keliais metodais: komponentų išvadus pervedant per lydvietės ertmę (senas metodas), plokštės paviršiuje prie lydmetaliu padengtos varinės plokštelės prilituojant komponento išvadus (naujesnis metodas). Naudojant pirmąjį metodą, kiekvienas naudojamas elektroninis komponentas turi plonus laidelius arba išvadus, kurie yra prakišami kiauriai lydvietes ir lituojami padengiant lydmetiu kitoje plokštės pusėje išlindusius išvadus. Kol komponentai nėra sulituoti, lydvietėse jie laikosi tik gravitacijos ir trinties pagalba. Naudojant antrąjį metodą, kiekvieno komponento atitinkamos formos išvadai yra tiesiogiai sujungti su laidininko takeliais. Lydmetalio pasta, kuri susideda iš klijų, fliuso ir lydmetalio, yra užtepama ant lydvietės, kad pritvirtintu komponentą litavimo metu. Dažniausiai lituojama konvekcinėse krosnyse. [4][6]

1.2 Galimos problemos PCB gamyboje

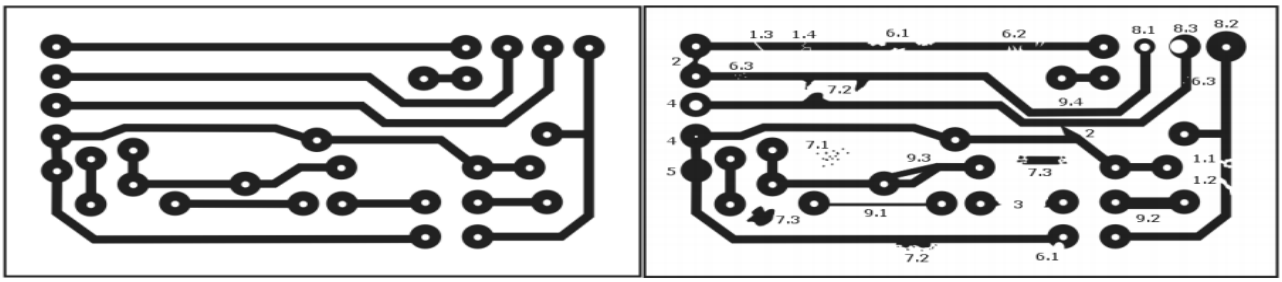
Spausdintinių montažinių plokščių (PCB) gamybojos metu gali atsistikti įvairiausių dalykų, dėl kurių galutinės gamybos stadijoje PCB yra išbrokuojama dėl defektų. Thibadeu pristatė apžvalgą apie PCB defektus bei jų atsiradimo priežastis. Kai kurias priežastis jis išskkyrė : blogi tarpai tarp takelių, takelių linijų nevienodumas, netikri aliarmai dėl dulkėtos aplinkos. Tokie defektai atsiranda dėl neteisingos pozicijos spausdinimo metu, temperatūros įtokos, dulkių, iškraipytų plokščių. Kita vertus, PCB yra brokuojama atsiradus tik kai kuriems defektams, smulkesni defektai PCB neįtakoja. [20]

Nors PCB gamybos pramonė vis tobulėja, tačiau dar ir šiomis dienomis gamybos metu atsiranda vis dar tokie patys defektai ir visą tai yra paspirtis plėtoti inspektavimo problemą. Greenberg (2006) pasiūlė U.S.A. patentą, kuris rėmėsi defektų inspektavimu, kai defektuota plokštė programinės įrangos pagalba yra lyginama su geru pavyzdžiu, kuris yra saugomas duomenų bazėje. Programinė įranga jaučia PCB geometriją, susidaro koordinates, kurias lygina su gero šablono kooridnatėmis. Naudojantis šia sistema buvo galima greitai atskirti geras PCB nuo tų, kurios yra defektuotos.[20][21]

PCB defektai gali būti klasifikuojami kaip visiškas defektas ir kaip potencialus defektas. Visiškas defektas priskiriamas tada, kai sukurta plokštė nebeatlieka jai skirtų funkcijų. Potencialus defektas priskiriamas tada, kai gali įvykti atsirasti klaidų, gedimų plokštės naudojimo metu.[20]

1.1 lentelė. PCB defektų klasifikavimas

Visiškasdefektas	1. Įtrūkimai	1.1 Lūžimas
		1.2 Nepilnas įlūžimas
		1.3 Įdrėskimai
		1.4 Įtrūkimai
	2. Trumpikliai	
	3. Trūkstama laidininko dalis	
	4. Blogas lydietės diametras	
	5. Lydvietės nebuvimas	
Potencialūs defektas	6. Dalinis atvirumas	6.1 Pelės įkandimas
		6.2 Smulkūs patrūkinėjimai
		6.3 Skylutės
	7. Melagingi defektai	7.1 Taškeliai
		7.2 Papildoma atšaka
		7.3 Papildomas takelis
	8. PCB pagrindo pažeidimai	8.1 Per mažai išėsdinta
		8.2 Per daug išėsdinta
		8.3 Netiksliai išėsdinta
	9. Pokyčiai tarp atspausdintų linijų	9.1 Per plonas takelis
		9.2 Per platus takelis
		9.3 Fiktyvus takelis
		9.4 Galimas trumpiklis



(a) (b)

1.1 pav. PCB schema be defektų (a). PCB schema su defektais (b).

1.3 PCB tikrinimo techologijos

PCB defektų inspektavimas yra esminis elementas bet kokiame elektronikos gamybos procese. Šiomis dienomis bet kokio lygio PCB plokščių gamyboje defektų inspektavimas yra būtinas.

Ankstyvoje PCB ar bet kokios kitos elektronikos įrangos gamybos proceso stadijoje, galutinio produkto apžiūra buvo daroma rankiniu būdu, defektų ieškodavo žmogus. Tačiau tobulėjant technologijoms, sudėtingėja ir pačios PCB, kieno pasekoje defektų inspektavimas, naudojantis žmogiškaisiai resursais, tapo nebepatikimas, todėl atsiranda didelė tikimybė, kad defektuota PCB pateks į kitą gamybos proceso etapą, kaip geras pusgaminis.[5]

Šiuo metu naudojamos tokios PCB defektų inspektavimo sistemos:

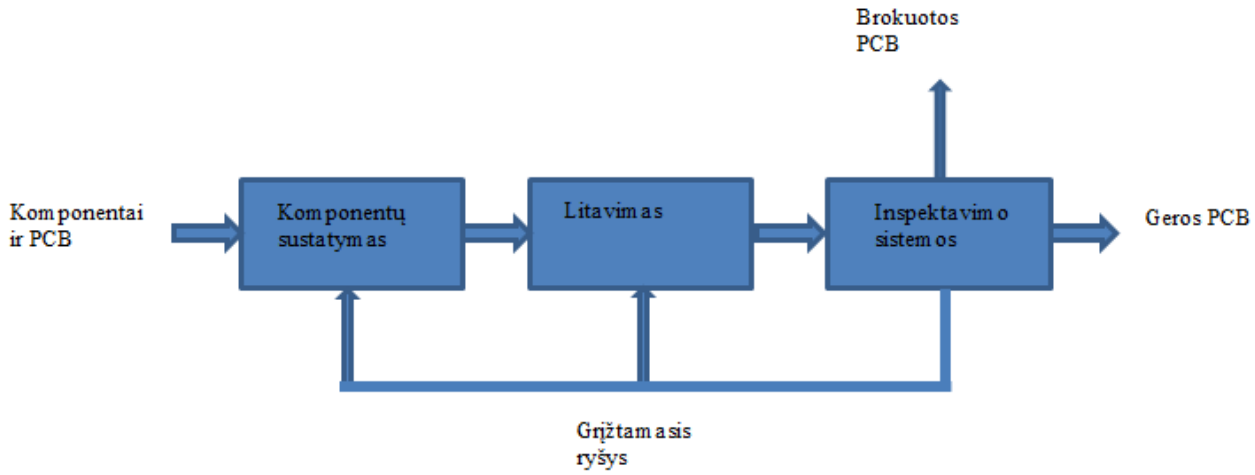
- **AOI** - automatinis optinis inspektavimas (angl. *Automatic Optical Inspection*). Automatizuotas optinis inspektavimas yra privilegijuotas PCB defektų inspektavimo metodas. Panaudota optinė sistema, kuri realiu laiku užfiksuotą atvaizdą lygina su sistemoje esančiu etaloniniu atvaizdu, sistema palyginusi abu atvaizdus išskiria esamus skirtumus ir suklasifikuoja neatitikimus. Ši PCB testavimo sistema yra plačiai naudojama, sistema yra nuolat tobulinama, kad veikimas taptų vis patikimesnis.
- **AXI** – automatinis rentgeno spindulių inspektavimas (angl. *Automated X-Ray Inspection*). Sudėtingėjant elektroninėms plokštėms, tankėjantys elementai apsunkina defektų inspektavimą, nes pasidaro sunku išskirti visas lydvietes. Kai kurie elektronikos elementai prilituojami taip, kad inspektuojant optiniu būdu yra matomi tik komponentų išvadai, tad neįmanoma nuspresti apie lydviečių kokybę. Norint eliminuoti šią problemą buvo pasitelkti rentgeno spinduliai (AXI), kurie lydviečių kokybę gali apspresti net tada, kai pati lydvietė yra uždengta su papildomomis mikroschemomis.
- **ICT** – kontaktinis testavimas (angl. In Circuit Test). PCB testavimas vyksta fiziniai prisilietimais, matuojant signalų pralaidumą takeliuose.

PagrindinėsPCB defektų inspektavimo sistemos paskirtys gamybos procese:

- **Gamybinių defektų pažymėjimas -** aiški PCB defektų inspektavimo sistemos funkcija yra pažymėti visus defektus, kurių būtų galima išvengti dar iki tikrinimo. Kuo ankstesnis

defektų radimas gamybo procese yra būtinas todėl, kad defektuotai PCB patekus į kitą gamybos proceso etapą, labai išauga gamybos broko kaštai.

- **Suteikti procesui grįžtamąjį ryšį** – taip pat svarbi defektų inspektavimo paskirtis yra suteitki gamybos procesui grįžtamąjį ryšį. Naudojant PCB apžiūros sistemas gali būti pastebimos bet kokios klaidos, suteikus grįžtamąjį ryšį procesui, tolimesnėje eigoje, visos klaidos atsiradusios dėl klaidingo proces yra eliminuojamos ankstyviausiose gamybos proceso stadijose.



1.2 pav. PCB gamybos proceso eiga

Iš 1.1 paveikslo galima matyti, kad PCB defektų inspekcijos metu suradus defektuotą plokštę yra suteikiamas grįžtamasis ryšys, dėl ko galima lengviau tobulinti gamybos procesą. Inspektavimo sistemos yra naudojamos taip pat ir po komponentų sustatymo, tiesa dauguma komponentų sustatymo įrenginių turi lokalias inspektavimo sistemas. Atliekant defektų inspektavimą dar prieš litavimą, galima lengvai pataisyti PCB.

1.3.1 AOI - automatinis optinis inspektavimas

Nepaisant daugybės patobulinimų, kurie būvo padaryti, šiuolaikiniai PCB takeliai yra išdėstyti labai painiai ir tankiai, todėl plokščių tikrinimas tampa daug sudėtingesnis, nei kad prieš kelis metus. Prasidėjus paviršinio montažo plokščių gamybai ir sekant tolimesniam komponentų dydžių sumažėjimui reiškia, kad plokštės yra labai kompaktiškos. Netgi vidutinio sudėtingumo plokštės turi tūkstančius lydviečių, kurios santykinai ir sudaro didžiausią defektų skaičių.[5]

Didėjant plokščių sudėtingumui atsiranda poreikis automatizuoti defektų inspekciją, nes šiomis dienomis raninė apžiūra nėra kokybiškas pasirinkimas. Kokybės inspektoriai pavargdavo, o dėl nuovargio atsirasdavo didelė tikimybė pro akis praleisti brokuotą plokštę. Šiuo metu rinka diktuoja sąlygas, dėl kurių gamyklos turi užtikrinti itin greitą ir patikimą defektų inspektavimo sistemą tam, kad atsirastų garantija, jog pagaminta plokštė yra itin aukštos kokybės. AOI, automatinė optinė inspekcija yra būtinas įrankis integruotos elektronikos testavimo srityje, kuri garantuoja, kad broko kaštai yra sumažinami iki minimalio sumos, nes brokas yra surandamas ankstyvoje gamybos stadijoje.[20][5]

Automatinė optinės inspekcijos sistema naudoja regimuosius metodus, kad aptiktų ir parodytų plokštėse esančius defektus. Sistema gali aptikti įvairius paviršutinius defektus, tokius kaip : *trumpikliai, takelių lūžimai, įbrėžimai, lydmetalio išretėjimas, lydviečių pakitimai, komponentų tinkamumą, komponentų sulitavimo kokybę, bei klaidingai sudėtus komponentus.* Sistema atlieka analogišką inpekciją, kokią atlikdavo ir žmogus, tačiau tai atlieka kur kas greičiau ir tiklsiau.[5]

Tai atliekama tyrinėjant plokštės paviršių. Plokštė yra apšviečiama keliais šviesos šaltiniais ir atvaizdui išgauti yra naudojama viena ar kelios aukštos kokybės kameros. Tokiu būdu AOI įrenginys išgauna inspektuojamos plokštės atvaizdą. Sistema apdoroja gautą atvaizdą ir palygina jį su įrenginyje sukauptomis žiniomis, kurios gautos apdorojus etaloninės plokštės atvaizdą. Naudojantis tokiu principu, automatinė optinės inspekcijos sistema tampa įgali inspektuoti ir pažymėti defektus ar įtartinas plokštės vietas. [20]

Dažniausiai naudojamos AOI naudojami metodai norint nustatyti ar plokštė yra brokuota, ar brokas yra toleruotinas:

- Gauto plokštės atvaizdo lyginimas su šabloniniu (etaloniniu) atvaizdu. Tikrinama plokštė yra palyginama su šabloniniu atvaizdu ir remiantis po palyginimo gautais rezultatais, nustatomas inspektuojamos plokštės statusas.
- Modelio lyginimas. Naudojant šį metodą AOI sistema išsaugo inspektuojamos plokštės informaciją ir ją lygina su etaloniniu atvaizdu.
- Statistinis modelio lyginimas. Šis metodas panašus į paminėtus aukščiau, išskyrus tai, kad jame naudojamas problemų sprendimas yra pagrįstas statistika.

AOI atvaizdo gavimas ir analizė

Vienas pagrindinių AOI sistemos elementų yra automatizuotas inspektuojamos plokštės atvaizdo gavimas naudojantis optine inspektavimo sistema. Gautas inspektuojamos plokštės atvaizdas yra apdorojamas AOI programinės įrangos. Egzistuoja nemažai plokštės atvaizdo gavimo būdų, kurių pasirinkimas tiesiogiai priklauso nuo tikrinamų plokščių sudėtingumo, kainos.[20]

Vaizdo išgavimo sistemą gali sudaryti ir viena kamera, tačiau, jei yra poreikis išgauti 3D atvaizdą, kamerų skaičius išauga iki kelių. Kameros pritvirtinamos ant automatizuotų mazgų, kurie pagal programinės įrangos adresavimą perkelia kamerą prie tam tikro inspektuojamos plokštės bloko.[20]

Nuo pasirinktos kameros tipo tiesiogiai priklausys ir gauti rezultatai.Greitis prieš tikslumą – tai yra esminis balansas, pagal kurį yra pasirenkama kamera:

- Srautinis vaizdas: vieno tipo kamera, kuri yra naudojama automatiniam optiniui inspektavimui, filmuoja inspektuojamą plokštę pravažiuodama pro šalį ir užfiksuodama visą plokštės rėmą ir puslaidininkius elementus. Iš nufilmuoto video suformuojamas stabilus plokštės atvaizdas, atliekamas tolimesnis atvaizdo apdorojimas. Šis būdas nėra labai tikslus, tačiau turi labai didelį greičio pranašumą.
- Nejudančio atvaizdo gavimas: atvaizdo gavimo metu, kamera būna arti inspektuojamos plokštės, gavus komandą yra užfiksuojama plokštės nuotrauka, tada ji yra apdorojama. Labai didelę svarbą šiame procese užima apšvietimas.

Kai yra pradedamas plokštės atvaizdo analizavimas, AOI sistema ieško konkrečių bruožų: komponento įstatymo vietos, komponento dydžio, komponento pavadinimo, plokštės markiracijos, fono splavos, atspindžio, lydviečių kokybės ir t. t.

AOI šviesos šaltiniai

Apšvietimas yra vienas iš esminių elementų AOI sistemoje. Pasirenkant atitinkamą apšvietimo šaltinį atsiranda galimybės daug efektyviau surasti skirtingų tipų defektus. Per pastaruosius metus, apšvietimo technologijoms padarius pažangą, defektų inspektavimo sistemose atsirado galimybė didinti ir mažinti pasirinktus plokštės atvaizdo segmentus, norint kuo geriau išskirti rastus defektus, nepakenkiant atvaizdo kokybei, kai tikrinimo greitis yra pakankamai didelis.[20][5]

Dauguma AOI sistemų turi nustatytą apšvietimo rinkinį, tačiau daugumai atvejų, apšvietimo rinkinio pasirinkimą nulemia inspektavimo operacijos sudėtingumas ir inspektuojamų plokščių sandara. Dažniausiai sistemos apšvietimas yra optimizuojamas pagal esamas sąlygas.[20]

Naudojami šviesos šaltiniai:

- Fluorescenscinis apšvietimas plačiai naudojamas AOI sistemose, kadangi sėkmingai apšviečia ir išryškina dažniausiai pasitaikančius defektus. Vienintelė problema naudojant tokio tipo apšvietimą yra tai, kad fluorescencinės lempos su laiku praranda savo šviesos intensyvumą, kieno pasekoje krenta ir inspektavimo kokybė.
- LED apšvietimas inspektavimo sistemose naudojamas dažniausiai. LED apšvietimas blanksta, kai jis naudojamas ilgą laiką, tačiau tai kompensuojama padidinant srovę. Naudojant LED apšvietimą galima keisti apšvietimo lygį, todėl šviesos diodai yra kur kas pranašesni nei fluorescencinės ar kaitrinės lemputės.
- Infraraudonųjų spindulių arba ultravioletinis apšvietimas naudojamas retai. Dažniausiai tokio tipo apšvietimas reikalingas surasti tam tikrus defektus, kurių neįmanoma aptikti naudojant fluorescencinį ar LED apšvietimą.

AOI sistemos programavimas

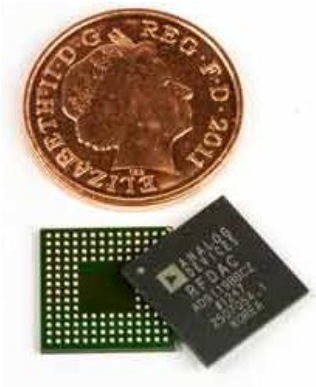
Tam, kad būtų galima patikrinti PCB plokštę naudojant optinį inspektavimą, nedefektuotos PCB plokštės duomenys turi būti saugomi AOI sistemoje. Pradinis duomenų apdorojimas ir funkcijų rašymas turi būti atliekamas labai tiksliai, norint teisingai aptikti visus defektus.[5]

AOI sistemos programavimo būdai:

- "Auksinės plokštės", kaip sistemos apmokymo priemonės naudojimas. Etaloninė plokštė yra nuskenuojama optiniu būdu, tada sistema atpažysta tam tikrus plokštės požymius (komponentus, litavimo taškus, takelius ir daugybę kitų aspektų). Siekiant užtikrinti pakankamą tikslumą, sistema turi būti apmokinama su keliomis "auksinėmis plokštėmis".
- Algoritmu paremtas programavimas. PCB duomenys yra pateikiami sistemai kaip kodas, sistema sugeneruoja savitą virtualią plokštę. Algortimo patikrinimui taip pat naudojamos etaloninės plokštės, tačiau jų reikia mažiau, nei apsimokinant "Auksinės plokštės metodu".

1.3.2 AXI - automatinis rentgeno spindulių inspektavimas

Automatizuota rentgeno spindulių inspektavimo sistema gali rasti beveik visus defektus, su kuriais susiduriama PCB gamyboje. Ši sistema dažniausiai naudojama po litavimo proceso. Pagrindinis privalumas lyginant su AOI sistema yra tai, kad ši sistema gali inspektuoti defektus, kurie yra "paslėpti" po papildomu elementų sluoksniu.[22]



1.3 pav. BGA plokštės dydžio palyginimas su moneta

AXI inspektavimo sistemos spidnuliai gali ne tik prasiskverbti kiaurai mikroschemas , bet ir gali suteikti vidinį lydvietės vaizdą. Tokiu būdu inspektuojami defektai, kurie gali atrodyti visiškai toleruojami, tačiau dėl vidinių lydvietės savybių, lydvietė būna defektuota. Tai reiškia, kad ši sistema gali pateikti papildomos informacijos apie lydvietes, kas gali užtikrinti, jog gaminys atitiks keliamus standartus. [22]

1.2 lentelė. AOI, AXI ir ICT sistemų galimybių palyginimas

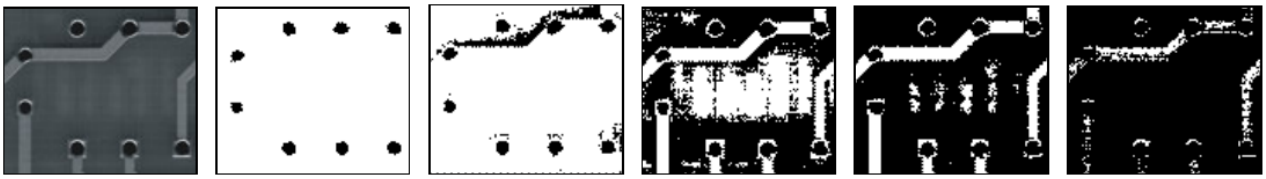
Pagrindinių PCB defektų identifikavimo galimybės AOI, AXI, ICT sistemose			
Litavimo defektai	AOI	AXI	ICT
Nutrūkusi el. grandinė	T	T	T
Lydmetalio trumpikliai	T	T	T
Nepakankamas lydmetalio kiekis	T	T	T
Lydmetalio porietumas	T	N	N
Lydmetalio perteklius	T	T	N
Lydmetalio kokybė	T	N	N
Komponentų defektai			
Lydvietės pakilimas	T	T	T
Trūkstamas komponentas	T	T	T
Netinkama komponento padėtis	T	T	T
Neteisinga komponento reikšmė	N	N	T
Brokuotas komponentas	N	N	T
BGA defektai			
BGA trumpikliai	Y	N	T
BGA nutrūkusi el. grandinė	Y	N	T

1.4 Panašūs darbai

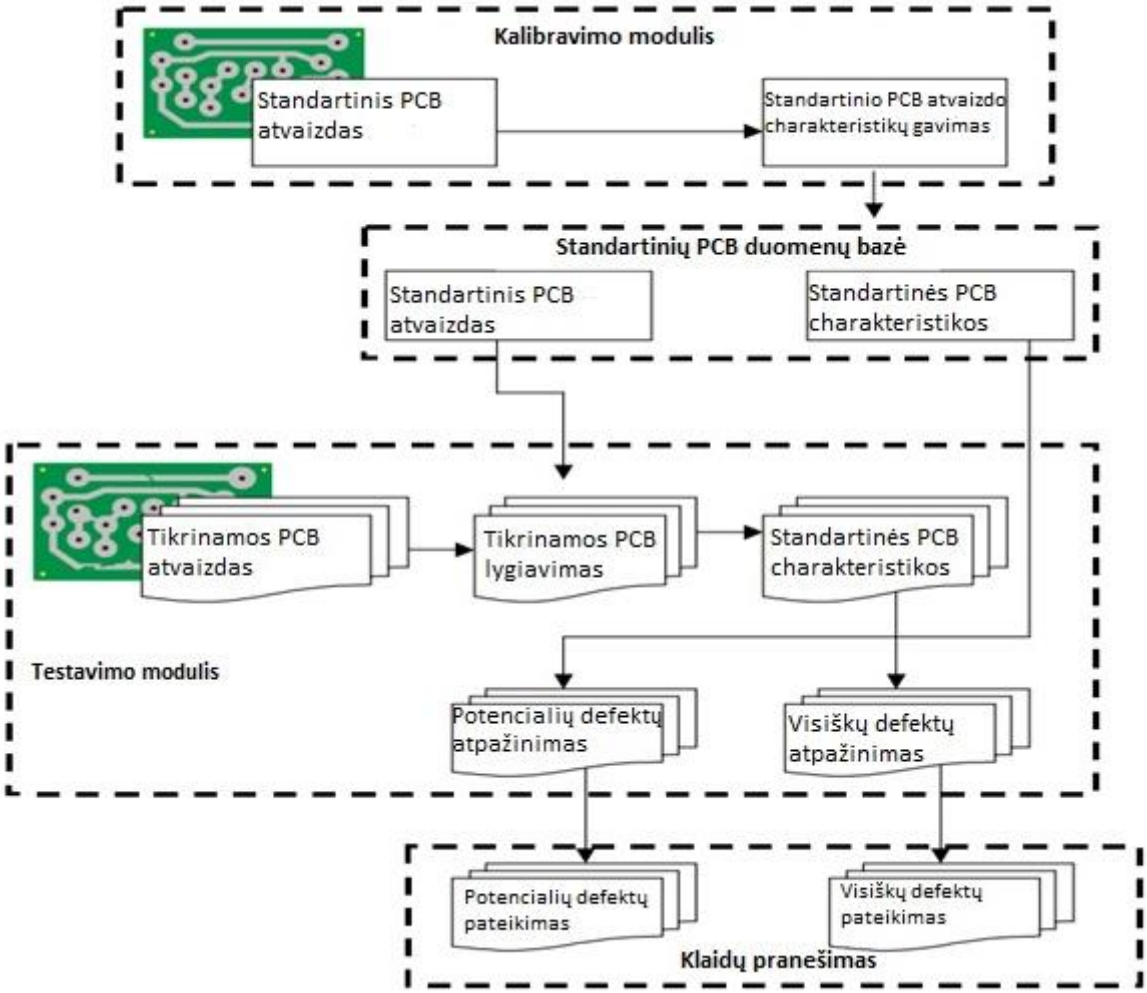
Kompiuterinė regos sistema išsivystė automatikoje taip, kad gali identifikuoti visišką defektą arba potencialų defektą ant plikos plokštės t.y. plokštės be papildomų el. komponentų. Sistema yra sudaryta iš dviejų pagrindinių modulių. Pirmasis modulis yra skirtas kalibravimui, o antrasis modulis skirtas testavimui. Kalibravimo modulyje etaloninė PCB geometrinės charakteristikos yra įsisavinamos ir išsaugomos duomenų bazėje. Gautas standartinis PCB atvaizdas ir jos charakterizuoti parametrai yra naudojami sistemos lygyje, kai gautas atvaizdas yra lyginamas su etaloniniu atvaizdu. Šiuo atveju yra labai svarbu, kad gautas PCB atvaizdas būtų paimtas analogišku rakursu kaip ir etaloninis atvaizdas, viskas turi būti viename lygyje.Bet koks pozicijų skirtumas gali reikšti klaidingą defekto aptikimą. Daugiau detalių apie plokščių lygiavimą, įskaitant atvaizdų lyginimo problemas galima rasti. Testavimo modulis yra padalintas į dvi dalis. Viena dalis identifikuoja potencialius defektus, kita dalis identifikuoja visiškus defektus remiantis Tatibana ir Lotufo algoritmu. Pažymima, kad siųloma technologija suteikia galimybę lengvai apdoroti iliustracijas, tačiau kai yra naudojamos tikros plokštės, atsiranda begalė sunkumų. Dėl šios priežasties atsirado naujos, inovatyvios technologijos ir pritaikyta teisinga metodologija ir tikroms PCB, sistemos schema parodyta 1.5 paveiksle.[14]

Etaloninis ir plokštės atvaidas yra paverčiamas į dvejatainius atvaizdus. Nors tai yra įprastas procesas, jo taikymas dar nėra išdirbtas ir kartais galutinis rezultatas būna visai ne toks, kokio yra

tikimasi. Dėl nevienodų apšvietimo salygų, beveik tampa neįmanoma atskirti laidininkus nuo paviršiaus.Todėl sistema turi būti sureguliuota, ji turi įgyti sąlygas kiekvienai plokščių kategorijai. Didžiausia problema yra „geriausio slenksčio“ faktorius. Stebint 1.4 paveikslą, suprantama, kad tiriamo paveisklo apdorojimas yra gana neblogas ir galima tikėtis nedefektuotos plokštės. Pavyzdžiui Gokturk (2005m) pasiūlė metodologiją kaip išspręsti šią problemą, kuri susideda iš modifikuotos „Gudraus kampų atpažinimo“ ir neprižiūrimų mokymosi algoritmų, kurie atskiria skirtingus PCB plokštės taškus.[15]

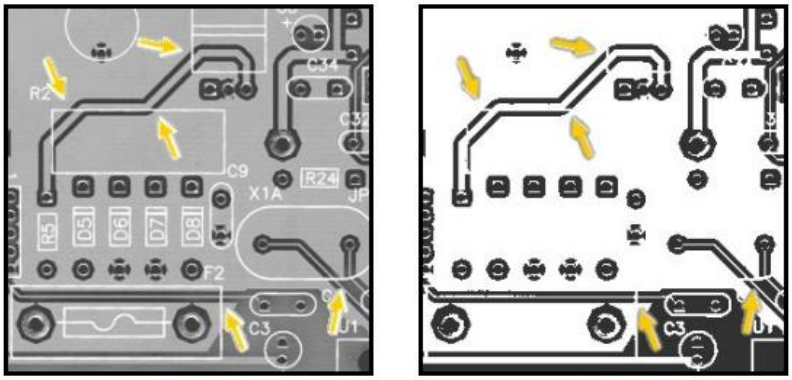


1.4 pav. PCB dalis (a). Slenkčio reikšmės lygios : (b)4, (c)50, (d)70, (e)80 ir (f)110.



1.5 pav. PCB inspektavimo sistema

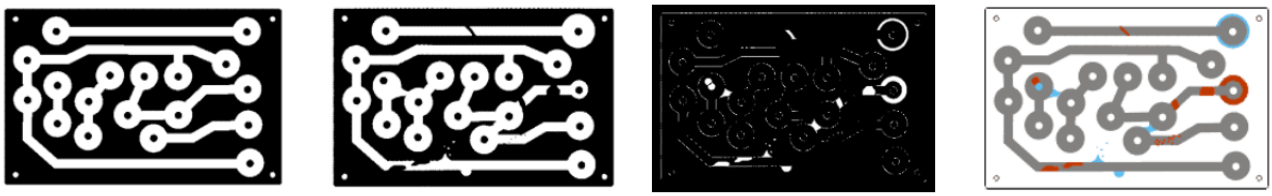
Sekanti, dažnai aptinkama problema prieš apdorojant atvaizdą yra informacijos atspausdinimas ant plokštės (1.6 pav.). Šiuo atveju, po paveikslo apdorojimo gali atsirasti klaidinga trūkusio takelio interpretacija. Taigi, atvaizdo apdorojimą rekomenduojama atlikti prieš spausdinimo procesą.[15]



(a) (b)

1.6 pav. (a) atspausdinta informacija gali įtakoti neteisingą analizė. (b) po atvaizdo apdorojimo vienas iš takelių atrodo pažeistas.

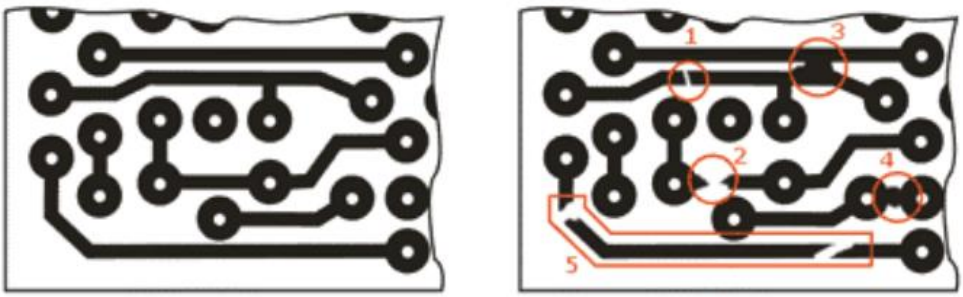
Sprendžiant šias problemas PCB atvaizdas yra lyginamas su etaloniniu PCB atvaizdu. Potencialios atpažinimo klaidos yra nustatomos iš tikrinamo atvaizdo atimant etaloninį atvaizdą. Jei, po atvaizdo atimties, tirkinamame atvaizde dar randama papildomų pikselių, sistema juo nuspalvina raudonai, jei po analogiško veiksmo papildomų pikselių atsiranda ir netinkamose vietose, sistema juos nuspalvina mėlynai (1.7 pav.). Pikselių trūkumas gali parodyti trūkusį takelį, išėsdinimus, per daug plonus takelius ar trumpą jungimą. Galime pastebėti, kad kai kurie defektai klasifikuojami kaip visiški.[16]



(a) (b) (c) (d)

1.7 pav. (a)Etaloninė PCB. (b) tikrinama PCB. (c) Dvejetainis atvaizdas. (d) Spalvotas atvaizdas.

Visiški gedimai yra identifikuojami iš naujo (jei yra poreikis) elektrinio ryšio konceptu. Elektroninių komponentų sujungimo su PCB tikslas yra tas, kad tai leidžia srovei tekėti norima linkme. Kadangi PCB plokštėje yra daug puslaidininkų, kurie yra sujungti, tai jei tikrinant yra ryšis, plokštė nėra išbrokuojama. Analizuojama inspektavimo sistema atpažysta takelių lūžimus (1, 2, 5), trumpus jungimus (3, 4) (1.8 pav.). [16]



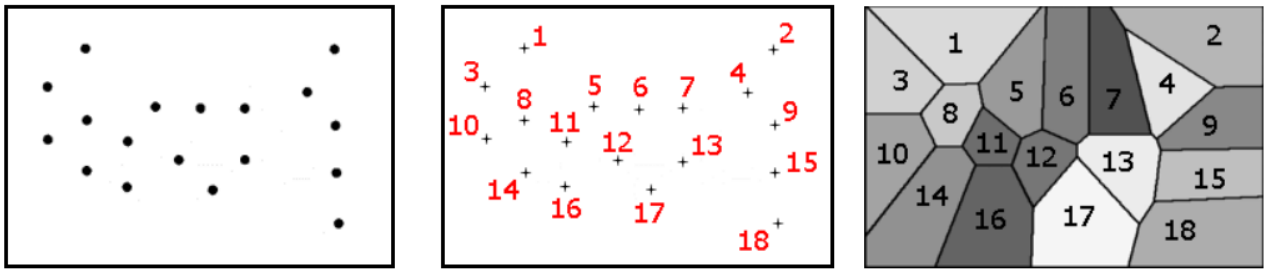
(a) (b)

1.8 pav. (a)Etaloninis atvaizdas. (b) Tikrinamas atvaizdas su visiškais defektais

Pirmą žingsnį sudaro tai, kad reikia atpažinti laidininkus ir skyles, tuomet sudaroma ryšių lentelė. Ši lentelė yra lyginama su etalonine ryšių lentele, gautos kalibraciniame modulyje. Toks pat metodas naudojamas gauti etaloninę lentelę ir tikrinamo atvaizdo lentelę.[16]

Pirmiausia, norint atpažinti skyles, reikia apdoroti PCB atvaizdą taip, kad dominuotų juoda ir balta spalvos. Naudojamas ryšių operatorius, kuris apdoroja baltas skyles. Skylių centrų koordinatės yra apskaičiuojamos ir išsaugomos lentelėje (1.9 pav.).[17]

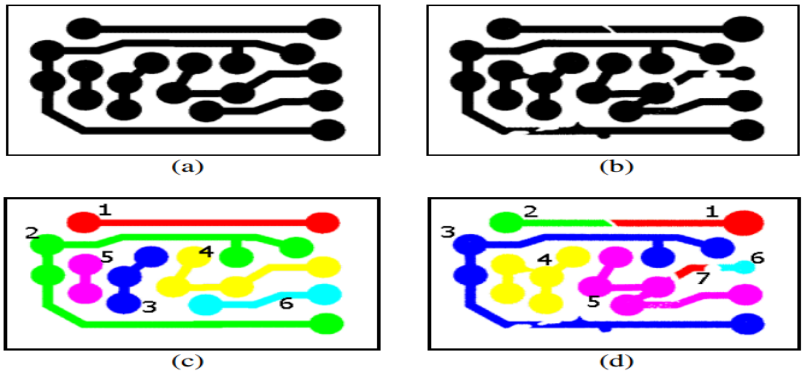
Lyginant kiekvienos skylės indeksą etaloniame ir tikrinamame atvaizde, galima pažymėti, kad nevisada vienodas skylės indeksas parodo vienodas skyles lyginamuose atvaizduose. Tam, kad išvengtu tokių neaiškumų, buvo įdiegta įtakos žemėlapių algoritmas. Įtakos žemėlapis yra grindžiamas etaloniniu PCB atvaizdu. Šis atvaizdas yra padalintas į skirtingas zonas, pagal minimalų atstumą tarp kiekvienos skylės centrinių koordinačių ir kievieno atvaizdo pikselių. Minimalus rezultatas atitinka skylės padėtį specifinėje zonoje. Susikirtimas tarp įtakos zonų yra neaiškus. [14][15]



(a) (b) (c)

1.9 pav. (a) Skylės ir jų (b) koordinatiniai indeksai (etaloninio atvaizdo). (c) Įtakos zonų žemėlapis.

Analizuojant laidininkus, skylės yra užpildomos juoda spalva. Tuomet 4-kontaktis operatorius) yra įdiegiamas į naują atvaizdą, gauti atvaizdai parodyti 1.10 paveikslo (c) ir (d) dalyse, kiekvienas laidininkas turi savo spalvą. Jei nor vienas laidininkas yra nutrūkęs, algoritmas jį pažymi skirtinga spalva (1.10 pav.)(spalvos aprašytos 1.3 lentelėje), kas padeda greičiau atpažinti potencialų defektą.[17]



1.10 pav. (a)Etaloninis atvaizdas. (b) Tiriamas atvaizdas. (c) ir (d) laidininkai išskirti spalvinėmis žymomis

1.3 lentelė. Spalvinis laidininko žymėjimas

Indeksavimo spalvos			
Etaloninis atvaizdas		Tikrinamas atvaizdas	
1	Raudona	1	Raudona
2	Žalia	2	Žalia
3	Mėlyna	3	Mėlyna
4	Geltona	4	Geltona
5	Rožinė	5	Rožinė
6	Žydra	6	Žydra
		7	Oranžinė

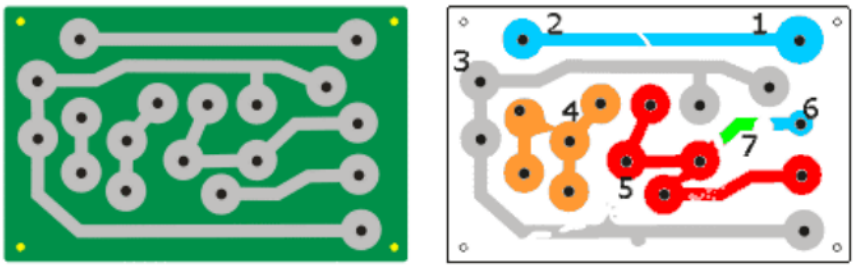
1.4 lentelė. Etaloninio PCB ryšiai, tikrinamo PCB ryšiai

Etaloniniai ryšiai		Tikrinamoatvaizdo ryšiai	
Laidininko indeksas	Lydvietės indeksas	Laidininko indeksas	Lydvietės indeksas
1	1	1	2
2	1	2	1
3	2	3	3
4	2	4	3
5	3	5	4
6	4	6	5
7	2	7	3
8	5	8	4
9	4	9	6
10	2	10	3
11	3	11	4
12	4	12	5
13	4	13	5
14	5	14	4
15	6	15	5
16	3	16	4
17	6	17	5
18	2	18	3

Naudojant įtakos zonų žemėlapį sutalpinti laidininkus ir skyles į tikrinamų plokščių indeksų lentelę, kur kiekvienas laidininkas sujungia dvi skyles. Jei ten yra koks nors mažas įtrūkimas, jis aiškiai pasimatys, kai sulyginsim su etaloninio ryšio indeksavimo lentele (1.4 lentelė). 1.5 lentelė rodo spalvų koduotę, kuri priklauso nuo atvaizdo rezultato. Rezultatai gauti lyginant etaloninę ryšių indeksų lentelę su tiriamo atvaizdo ryšių lentele (1.11 pav.). 1.10 paveikslas parodo rezultatus, atsižvelgiant į spalvų kodą pateikta 1.5 lentelėje.[14]

1.5 lentelė. Spalvų kodai

Spalvos kodas		
Laidininkas	Spalvos reikšmė	Spalva
Nėra klaidų	0	Juoda
Nėra lydvietės	4	Žalia
Įtrūkimas	6	Mėlyna
Trumpiklis	8	Oranžinė
Įtrūkimas ir trumpiklis	10	Raudona



(a) (b)

1.11 pav. (a)Etaloninis PCB. (b) Ištirtos plokštės galutinis rezultatas

2. METODINĖ DALIS

Metodinėje dalyje yra aprašomas PCB defektų inspektavimo metodas. Defektų inspektavimo metodas realizuotas naudojantis „*Matlab*“ programiniu paketu.

Matlab buvo sukurta kompiuterių analitiko Clev Moler 1970m. Nuo tada ši programa buvo tobulinama ir šiuo metu virto sėkmigu komerciniu produktu, taikomu tiek mokslo technologijoms, tiek studijoms. *Matlab* – tai dviejų angliškų žodžių Matrix ir Laboratory trumpinys, parodantis taikomą esminį skaičiavimų būdą, grįstą su matricomis atliekamais veiksmais. Esminis šios programos duomenų elementas yra matrica, kurios tikslaus dydžio nurodyti nereikia. Programa papildoma specifinių taikomųjų sprendinių bibliotekomis, kurias galima naudoti kaip „juodąsias dėžes“, bet kartu ir mokytis analizuoti jų kodą.[1]

Naudojantis aprašytu metodu PCB plokštėje inspektuojami dažniausiai pasitaikantys defektai: *takelio trūkimai, trumpikliai, trūkstamos laidininko dalys, blogas lydviečių diametras, fono klaidos.*

Sukurtas metodas skirtas taikomojo uždavinio sprendimui. Vaizdo apdorojimas (toliau - inžinerinis uždavinys) ir jo savybių išgavimas siejamas su nuosekliu, struktūrizuotu uždavinio sprendimu. Inžinerinis uždavinys turi būti sprendžiamas metodiškai, taikant atitinkamą algoritmą, žingsnis po žingsnio, artėjant prie sprendinio. Skaičiavimais grindžiamo uždavinio sprendimo procesas aprašomas šiais pagrindiniais žingsniais:

1. Problemos apibrėžimas ir apibūdinimas.
2. Skaitinio metodo kūrimas uždaviniui spręsti.
3. Skaitinio metodo įgyvendinimas.
4. Sprendimo patikra ir sprendimo įvertinimas.

2.1 Problemos apibrėžimas

Gaminant spausdintinio montažo plokštes (toliau - PCB), gaminimo proceso metu, visada yra tikimybė, kad PCB bus defektuota. Greičiausias būdas patikrinti PCB kokybę yra vizualinė inspekcija, kadangi nėra jokio kontaktinio tikrinimo, PCB kokybę nustatoma kompiuteriniu algoritmu, kuris identifikuoja galimo defekto vietą plokštėje ir aspkaičiuoja jo įtaką plokštei.

PCB defektai turi būti identifikuojami tiksliai, kadangi ne visi egzistuojantys defektai gali pakenkti plokštei, dalis defektų yra toleruojami ir kokybei nepakenkia. Dalis defektų, kurie yra kritiniai, yra vos pastebimi, todėl uždavinys tampa sudėtingas. Sprendžiant uždavinį reikia atkreipti dėmesį atvaizdo metriką, kadangi tikrinamų PCB atvaizdai negali būti identiški, nes visada

egzistuoja maža atvaizdo padėties, jo užfiksavimo momentu, paklaida dėl atvaizdo fiksavimo įrangos techninių nuokrypių.

Defektai turi būti suklasifikuoti į atskiras klases: *trumpikliai, takelio trūkimai, lydviečių pakitimai, takeliu toleruotini įtrūkimai (pelės įkandimas), sienos klaidos.*

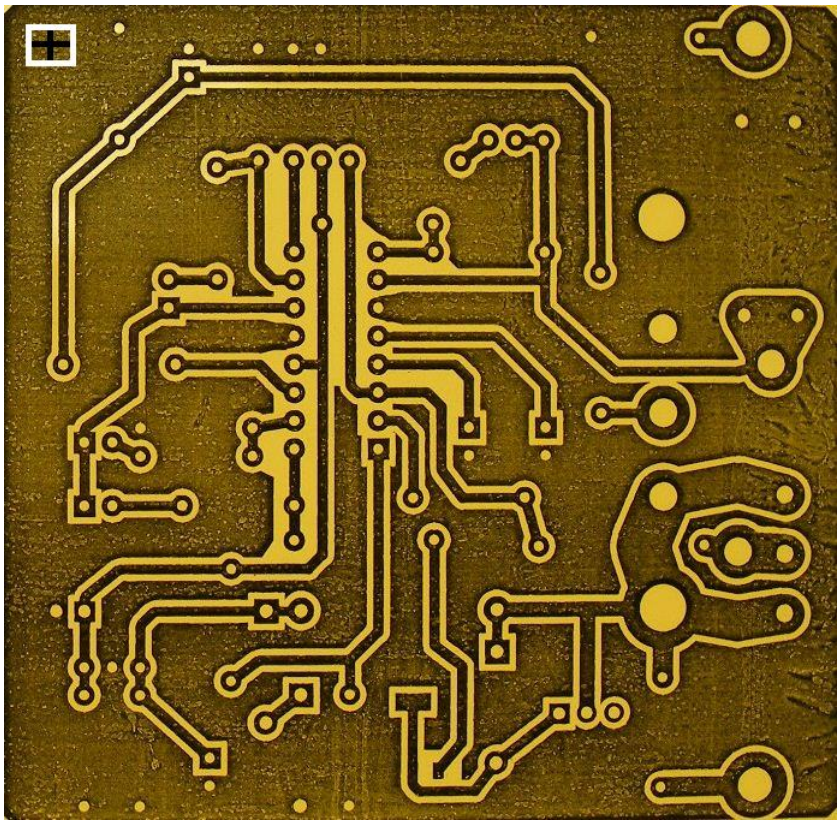
PCB defektų inspektavimo sistemos teisingo funkcionavimo patikrinimui pasirinktas spalvotas (RGB) PCB atvaizdas. Spalvotas atvaizdas turi būti apdorojamas, norint tesingai identifikuoti defektus. Atvaizdo filtravimas nuo triukšmo yra būtinas, nes identifikuojant defektus atsirastų

2.2 Skaitinio metodo kūrimas uždaviniui spręsti

PCB defektų inspektavimo sistemos veikimas pagrįstas etaloninės ir defektuotos plokštės palyginimo principu. Apdorojus etaloninės ir defektuotos plokščių atvaizdus, jie yra palyginami. Po palyginimo išskiriami defektuotos plokštės taškai, kurie neegzistuoja etaloninėje plokštėje.

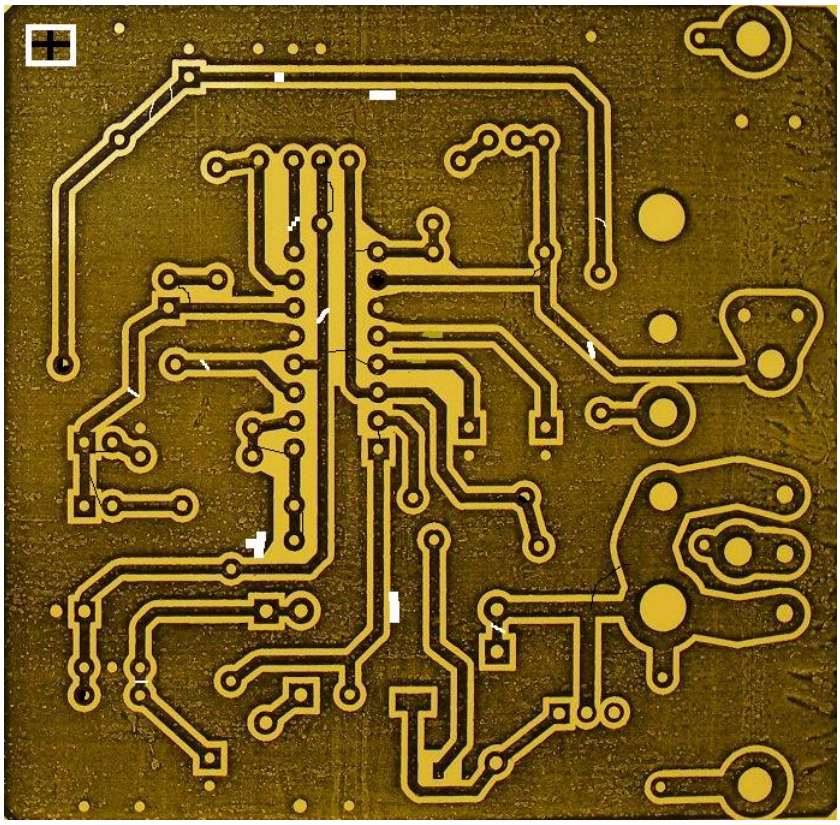
PCB defektų inspektavimo sistemos kūrimo metu buvo atsižvelgta į tokias problemas: *tikrinamo atvaizdo metrika (pasikreipimas etaloninės plokštės atžvilgiu), triukšmo filtravimas nepaliečiant smulkių defektų.*

Prieš pradedant kurti skaitinį metodą uždaviniui spresti, pasirenkama etaloninė (Pav. 2.1) ir defektuota (2.1 pav.) plokštės.



1.1 pav. Etaloninė plokštė

Etaloninės plokštės atvaizdas gautas nufotgrafavus PCB. Fotografijai (atvaizdui) išgauti buvo skaitmeninis fotoaparatas. Atvaizdo dydis – 693x678 pikselių.



1.2 pav. Defektuota plokštė

Defektuotos PCB plokštės atvaizdas yra analogiškas etaloninei plokštei. Defektai sukurti redaguojant atvaizdą su *windows 10* platformoje siūlomu redagavimo įrankiu. PCB atvaizdas redaguotas taip, kad vienoje plokštėje būtų visų ieškomų defektų rūšys. Redaguojant atvaizdą atsižvelgta į tai, kad realybėje vienos rūšies defektai būna įvairių dydžiu. Vieni defektai būna matomi ir plika akimi, kadangi defektas apima didesnę dalį plokštės, kiti - mažesni defektai plika akimi nepastebimi, tad reikia, kad juos identifikuotų inspektavimo sistemos.

Takelio lūžimo defektai realizuoti stengiantis nenukrypti nuo realių defektų, kurie atsiranda PCB gaminimo proceso metu. Takelio lūžimo defektai sudaryti taip, kad jie nebūtų lygiagretūs x ar y ašims. Dalis takelio lūžimo defektų tėra ~1 pikselio pločio, kas garantuoja, kad sukurtai sistemai radus tokio dydžio defektą, bus surasti visi takelių lūžimo defektai.

Trumpiklio defektai taipogi realizuoti stengiantis nenukrypti nuo realių defektų. Trumpikliai nėra lygiagretūs x ir y ašims, trumpiklio juostos plotis ~1 pikselį. Lydviečių defektai realizuoti užtušuojant dalį lydvietės ertmės. Toleruotini takelių defektai realizuoti ištrynus dalį takelio ir jį užpildžius fono spalva.

PCB defektų inspektavimo sistema sudaryta iš kelių etapų:

1. Testuojamų atvaizdų apdorojimas
2. Testuojamų atvaizdų objektų metrikos palyginimas
3. Testuojamų atvaizdų išvalymas nuo triukšmų
4. Atvaizdų palyginimas
5. Po palyginimo rastų skirtumų/defektų klasifikavimas
6. Defektų išskyrimas
7. Rastų defektų surašymas į registrą

2.2.1 Testuojamų PCB atvaizdų apdorojimas

Vaizdas gali būti apibrėžiamas, kaip dvimatė funkcija f (x, y), kur x ir y - taško koordinatės, o amplitude *I* lygi skaisčio reikšmei tame taške. Kai x, y ir amplitudės reikšmės yra baigtinės, diskretaus kiekio, galima teigti, kad vaizdas yra skaitmeninis. Skaitmeninių vaizdų apdorojimo srityje, norint perdirbti pirminius vaizdus, naudojami kompiuteriniai apdorojimo algoritmai. Pažymima, kad skaitmeninį vaizdą sudaro baigtinis elementų skaičius, kurių kiekvienas turi savo konkrečia vietą ir reikšmę. Minėti elementai yra nurodomi kaip *paveikslo elementai, vaizdo elementai, pikseliai, subpikseliai.* Dažniausia naudojamas terminas yra – pikselis.[11][12]

Matymas yra pats pažangiausias mūsų pojūtis, todėl nenuostabu, kad vaizdai užima svarbiausią rolę suvokime. Kaip bebūtų, skirtingai nuo žmonių, kurie turi ribotas elektromagnetinio spektro regėjimo juostų matymą, vaizdo apdorojimo įrenginiai apjungia ir gali apdoroti beveik visą elektromagnetinių bangų spektrą, nuo gamos iki radijo bangų. Įrenginiai gali apdoroti vaizdus, kurie sugeneruoti išorinių įrenginių (fotoaparatų, kamerų) pagalba, ko žmogus jau negali atlikti. Tai apima ultragarsą, elektroninės mikroskopijos ir kompiuteriu sukurtus vaizdus. Taigi, skaitmeninis vaizdų apdorojimas apima platų signalų ir vaizdų apdorojimo asortimentą.[12]

Vis dar nėra tokio bendro susitarimo tarp autorių, kur pasibaigia vaizdo apdorojimas pasibaigia, o kur prasideda vaizdo analizė ir kompiuterinis apdorojimas. Kartais vaizdo apdorojimas nusakomas kaip disciplina, kurioje tiek įėjime, tiek išėjime, apdorojimo proceso metu, yra vaizdai. Yra sritys, pavyzdžiui, kompiuterinė rega, kurios pagrindinis tikslas yra imituoti žmogaus regą, įskaitant mokymąsi ir išvadų darymą, remiantis vizualiniais įėjimais. Tokia sritis yra dirbtinio intelekto (DI) šaka, kurio pagrindinis tikslas yra pamėgdžioti ir bandyti atkartoti žmogaus intelektą. Kol kas DI yra nedaug išvystyta, vystymosi progresas buvo daug lėtesnis nei buvo tikėtasi. Vaizdo

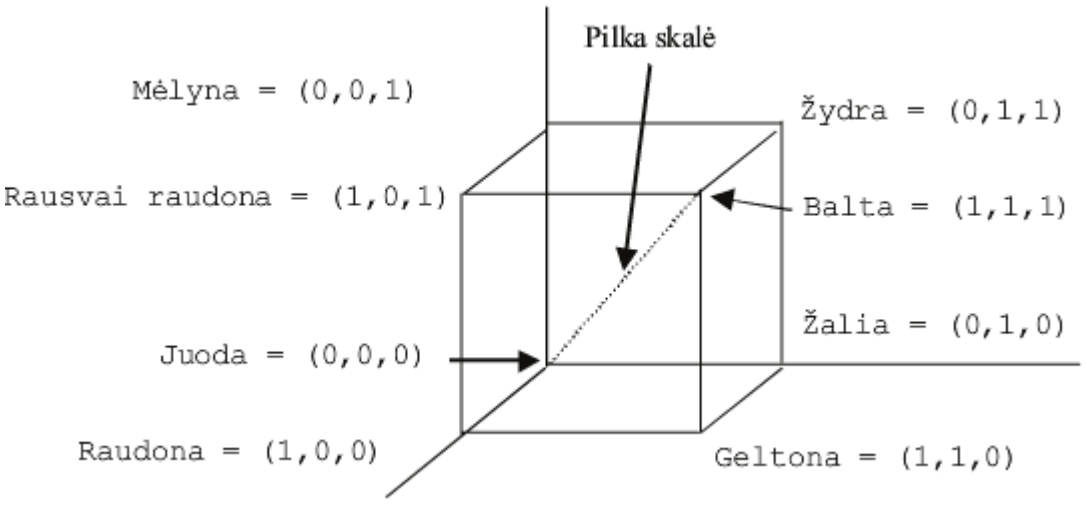
analizės terpė (taip pat vadinama vaizdo supratimu) yra tarp vaizdo apdorojimo ir kompiuterinės regos terpių.[13]

Nėra aiškių kontinuumo ribų nuo vaizdo apdorojimo viename gale, iki kompiuterinės regos kitame. Kaip bebūtų, yra viena naudinga paradigma nusakanti tris kompiuterinio apdorojimo lygius: žemo, vidutinio ir aukšto lygio procesai. Žemo lygio procesas apima paprasčiausius veiksmus taikomus vaizdams, t. y. triukšmo pašalinimas, kontrasto derinimas ir vaizdo paryškinimas. Žemo lygio procesas charakterizuojamas faktu, kad tiek įėjime, tiek išėjime yra vaizdai. Vidutinio lygio procesas, apdorojant atvaizdus, apima įvairias užduotis, tokias kaip vaizdo segmentavimas (vaizdo padalinimas į atskirus regionus), vaizdo glaudinimas,norint juos sumažinti į formą, tinkamą apdoroti kompiuteriu, atskirų detalių atpažinimui. Vidutinis lygis charakterizuojamas faktu, kad įėjime yra vaizdas, o išėjime yra išskirti tam tikros to vaizdo dalys (skeletas, regionas, kontūras). Galiausiai, aukščiausio lygio apdorojimas apima „savaiminį supratimą“ suprantant ir atpažįstant matomą vaizdą, naudojantis vaizdo analize ir atitinkamomis funkcijomis aprašyti kompiuterinei regai.[13]

Vaizdų apdorojimo programose pustoniai vaizdai saugomi kaip matricos, o spalvotieji – kaip daugiamačiai masyvai, pavyzdžiui, sudaryti iš trijų vienodo dydžio matricų, kurių vienoje yra raudonos spalvos dedamosios taškų reikšmės, o kitose atitinkamai – žalios ir mėlynos.

Testuojami PCB atvaizdai yra spalvoti (RGB spalvų paletė), todėl prieš pradedant atvaizdų apdorojimą juos reikia suskaitmeninti (paversti į dvinarį atvaizdą).

RGB spalvų paletė sudaryta iš trijų pirminių spalvų: R (angl. *red*) – raudonos, G (angl. *green*) – žalios ir B (angl. *blue*) – mėlynos. Maišant šias pirmines spalvas yra gaunama bet kokia kita spalva. Dažniausiai RGB spalvos vaizduojamos kaip kaip kubo kraštinės (2.3 pav.).[18]



2.3 pav. RGB spalvų kubas

Kai visos dedamosios yra lygios 0, gaunama juoda spalva, o kai maksimalios reikšmės – balta. Maišant visų spalvų po lygiai, gaunami pilki tonai (ang. *grayscale*).[7]

RGB yra patogi technikoje – daugelis vaizdo įtaisų veikia šios paletės pagrindu. Tačiau skalė nėra universali – ji netinka vaizdams apdoroti. Priežastis – R, G, B dedamosios tarpusavyje labai koreliuoja, o tai sukelia problemų naudojant vaizdo apdorojimo ir analizės algortimus. Pavyzdžiui, histogramos tankiui išlyginti geriau naudoti HIS skalę, kurioje yra spalvų intensyvumai. Daugeliu atveju vaizdams apdoroti yra reikalingi ne spalvoti, o juodos ir baltos spalvų su pustoniais vaizdai.[7]

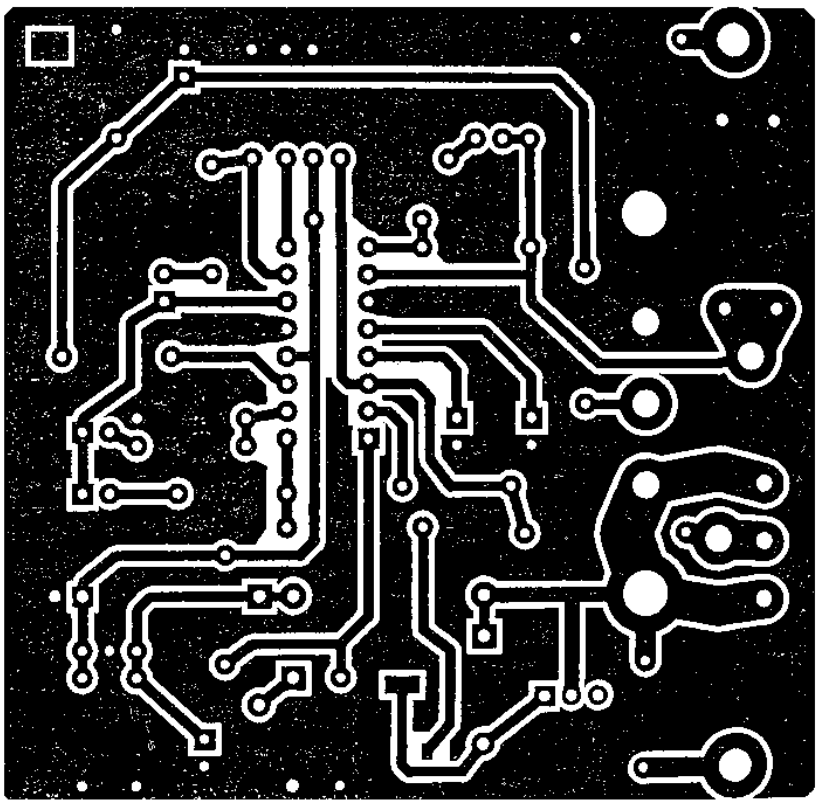
Norint perversti spalvotą vaizdą į juodai baltą reikia pasinaudoti formule:

$$JB = 0{,}333 \times R + 0{,}333 \times G + 0{,}333 \times B \qquad (2.1)$$

Jei vaizdas yra NTSC standartų, galima taikyti šiek tiek pakoreguotus koeficientus:

$$JB = 0{,}299 \times R + 0{,}587 \times G + 0{,}114 \times B \qquad (2.2)$$

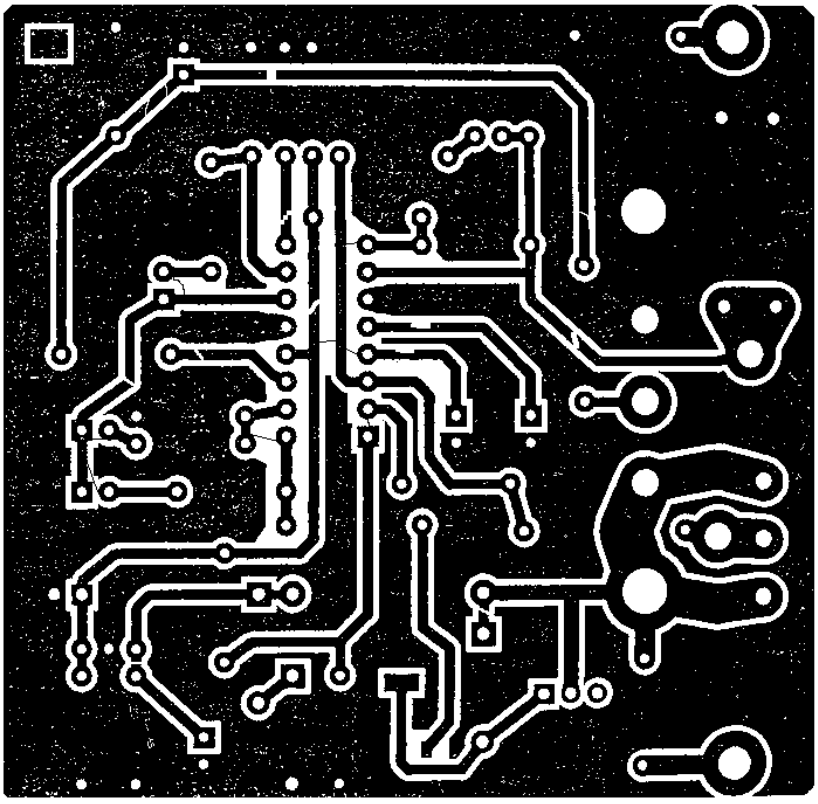
Atlikus atvaizdų skaitmenizavimą, gautuose atvaizduose (2.4 pav.) ir (2.5 pav.) matoma daug triukšmo.



2.4 pav. Triukšmingas etaloninis atvaizdas po skaitmenizavimo

Norint atlikti tolimesnius etaloninės ir defektuotos plokščių atvaizdų apdorojimo veiksmus, kuriamas triukšmo valymo filtras, kuris optimaliai pašalins visus nereikalingus triukšmus.

Vaizdų filtravimas dažniausiai naudojamas pradiniam vaizdui pakeisti į tokį, kuris atitiktų norimą spalvą matomoms smulkioms detalėms vaizde naikinti, norimiems vaizdo elementams išryškinti arba iškraipytiems vaizdams rekonstruoti. Iškraipytųjų vaizdų rekonstrukcijos metodai grindžiami iškraipymų modeliavimu ir atvirkštinių iškraipymams veiksmų atlikimu vaizdams atstatyti. Atliekant tokią rekonstrukciją svarbu nustatyti vaizdo kokybės kriterijus, kuriais remiantis sprendžiama, ar vaizdas yra atstatytas, ar ne. Vaizdo kokybės kriterijai yra euristiniai ir grindžiami žmogaus akies savybėmis.



2.5 pav. Triukšmingas defektuotas atvaizdas po skaitmenizavimo

2.2.2 Filtro kūrimas

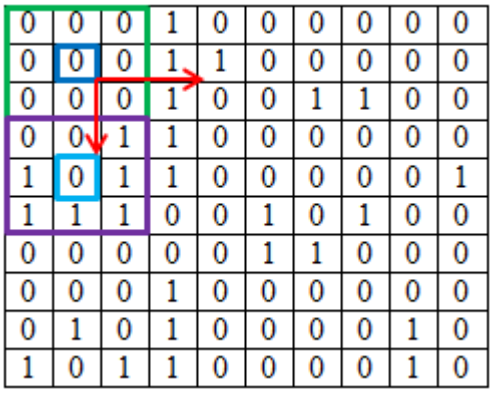
Kadangi, po atvaizdo suskaitmeninimo, gautame rezultate yra ganėtinai nemažai triukšmingų pikselių, todėl prieš pradedant tolimesnę apdorojimo seką, atvaizdą reikia išfiltruoti. Filtruojamo atvaizdo dydis – 693x678 pikselių.

Filtravime, kaip ir defektų inspektavime, naudojama 3x3 pikselių tikrinimo matrica (2.6 pav.).

0	0	0
0	●	1
0	0	1

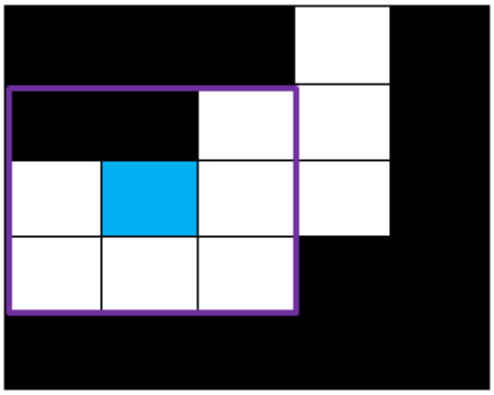
2.6pav. 3x3 matrica

Naudojant 3x3 tikrinimo matricą, patikrinamas kiekvienas atvaizdo pikselis. Tikrinimas pradedamas nuo atvaizdo masyvo antro stulpelio ir antros eilutės (2.7 pav.), kadangi tikrinamos 3x3 matricos vidurinis pikselis. Tikrinama, ar vidurinio pikselio (2.6 pav.) kaiminystėje esančių pikselių reikšmės, kiekvienas rastas baltas pikselis sumuojamas. Jei po patikrinimo, 3x3 matricoje baltų pikselių yra ≤ 4, balti pikseliai priskiriami triukšmui. Po priskyrimo triukšmui, balti pikseliai yra invertuojami į juodus pikselius. Jeigu tikrinant 3x3 matricoje baltų pikselių yra >4, tai suprantame, kad tai yra galima klaida, baltų pikselių neinvertuojame.



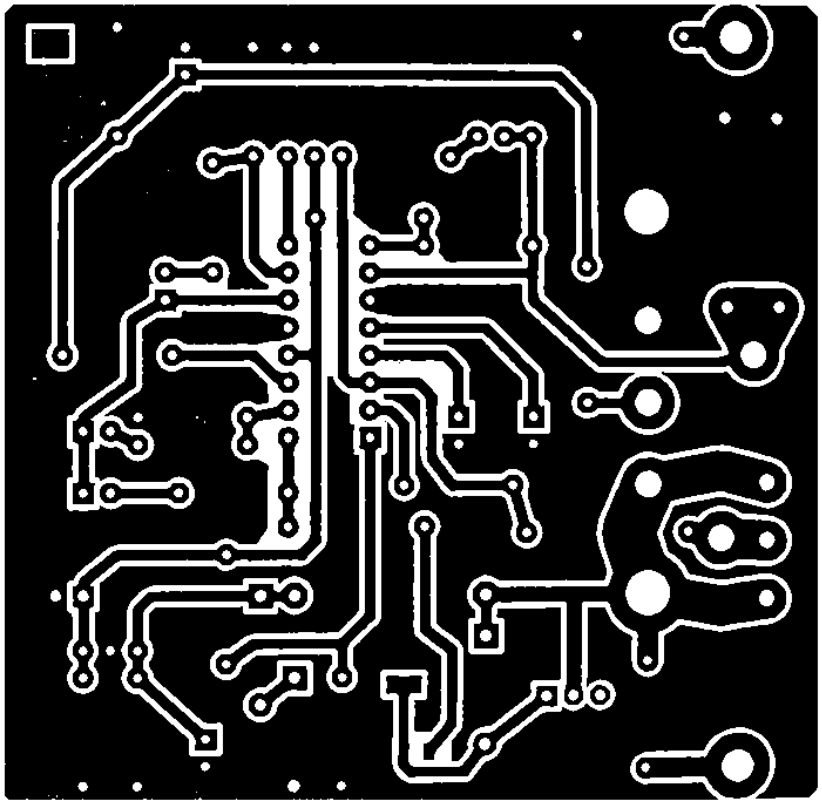
2.7 pav. 10x10 masyvo pikselių reikšmių tikrinimas (mėlynas langelis – tikrinimo pradžia, žalias ir violetinis langeliai – 3x3 tikrinimo matricos, žydras langelis – defekto detektavimo zona, rodyklės nurodo tikrinimo kryptį)

Toleruojamų baltų pikselių 3x3 tikrinimo matricoje kiekį galime koreguoti nuo 1 iki 8. Tačiau optimaliausiai atvaizdas išvalomas nuo triukšmo ties 4 baltų pikselių toleravimo riba. Kaip parodo 2.7 paveikslas, skaitmenizuoto atvaizdo masyvo tikrinimas prasideda ties antra eilute ir antru stulpeliu. Tikrinamame masyve, tikrinimo matricos viduriniąjam nariui pasislinkus į žydrai apibrėžtą zoną (2.7 pav.), pagal užduotą sąlygą yra detektuojamas galimas defektas, kadangi 3x3 matricoje iš 9 pikselių, 6 pikseliai yra balti.

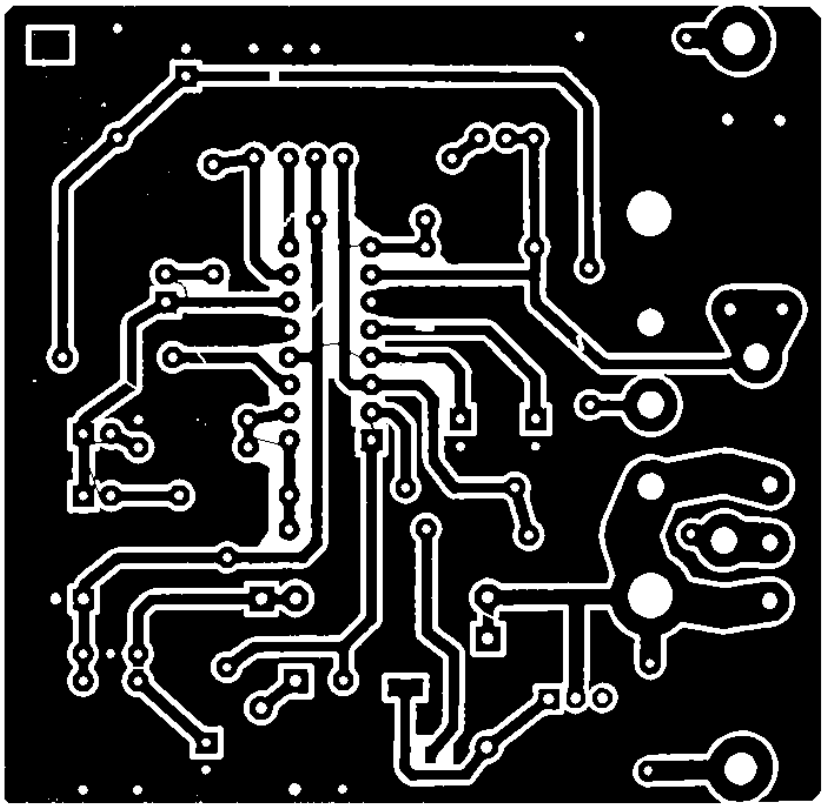


2.8 pav. Detektuotas defektas iš 2.7 paveikslo

Atlikus filtravimą tikrinamų PCB atvaizdams matyti, kad etaloninės plokštės atvaizde (2.9 pav.) beveik visas buvęs triukšmas yra pašalintas, o tikrinamos plokštės atvaizde (2.10 pav.) triukšmas pašalintas analogiškai kaip ir etaloninės plokštės atvaizde, tačiau su triukšmu kartu buvo išfiltruoti ir mažiausi takelių trūkimo defektai. Defektai takeliuose išfiltruojami, kadangi jie yra ~1 pikselio pločio ir nėra lygiagretūs tikrinimo matricos judėjimo x ir y ašims. Kadangi filtruojant tikrinamos plokštės atvaizdą išsifiltravo ir dalis takeliuose esančių defektų, buvo sukurtas analogiškas filtras, tačiau su juo buvo filtruojami tik takeliai.

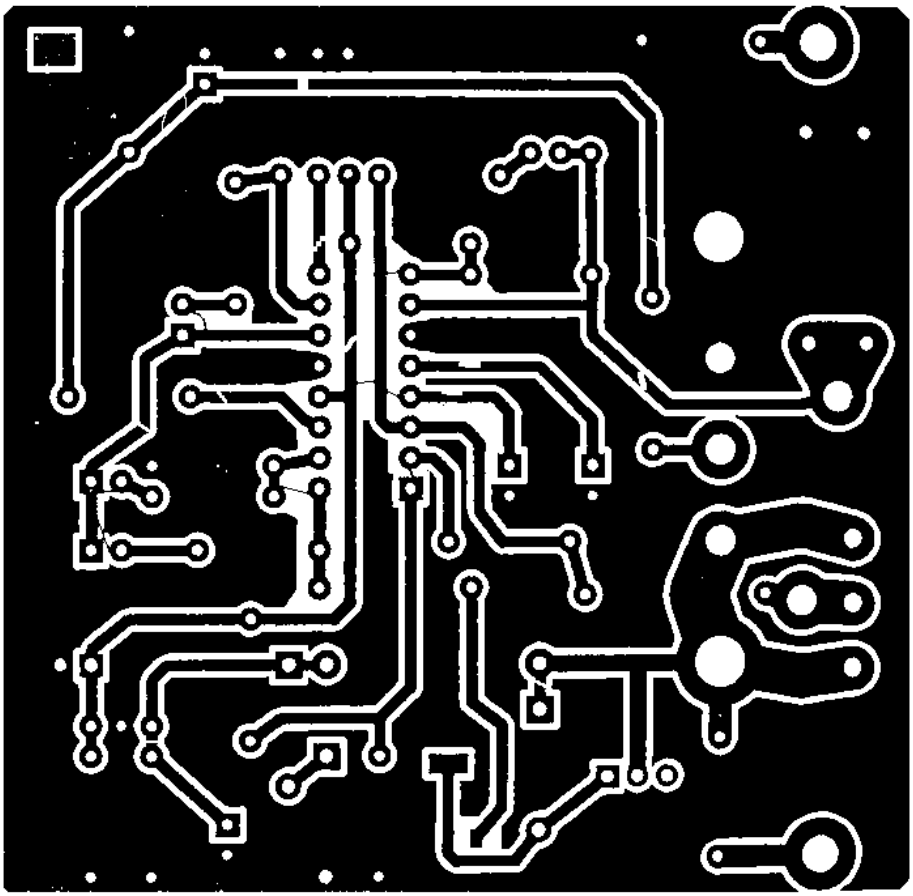


2.9 pav. Etaloninės plokštės atvaizdas po filtravimo



2.10 pav. Tikrinamos plokštės atvaizdas po filtravimo

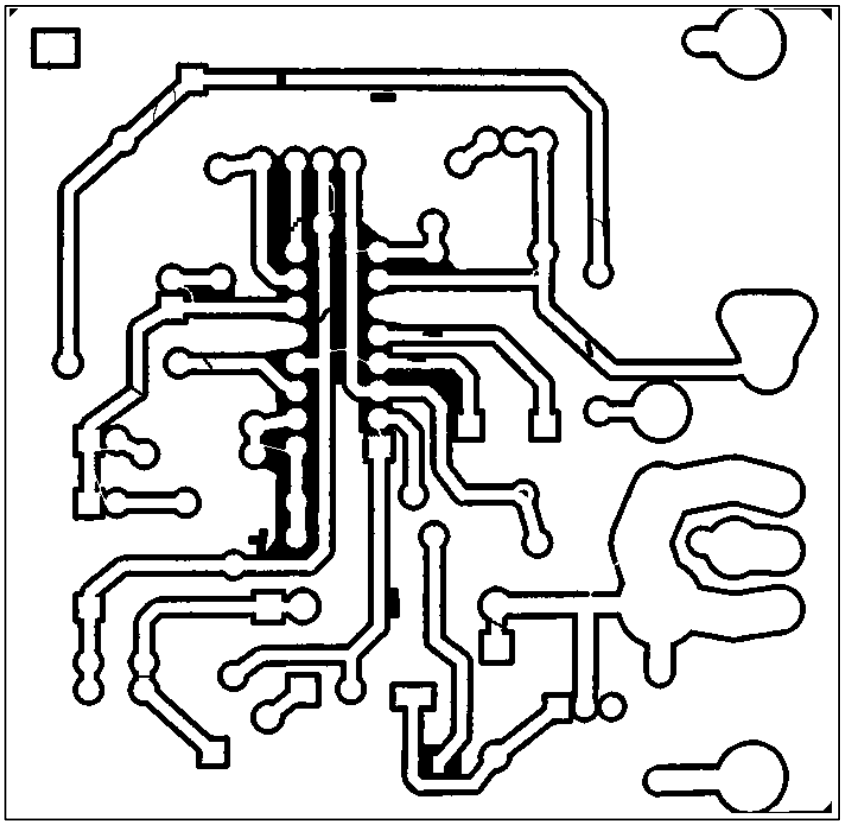
Kadangi takelius reikia filtruoti su aukštensne baltų pikselių toleravimo riba, reikia išskirti takelius iš tikrinamos plokštės atvaizdo. Tai realizuojama invertuojant etaloninį atvaizdą ir suradus visus jungius objektus. Pavieniai objektai, kurie nėra apvalūs (lydvietes) ir kurie nėra didžiausio ploto yra išskiriami. Kitaip tariant – takeliai iškerpami ir iš jų padaroma kaukė, kuri sutapatinama su tikrinamu atvaizdu, filtravimo metu. Išvalius tikrinamo atvaizdo fono triukšmus, kaukė yra pašalinama ir atvaizdas filtruojamas su 7 baltų pikselių toleravimo riba.



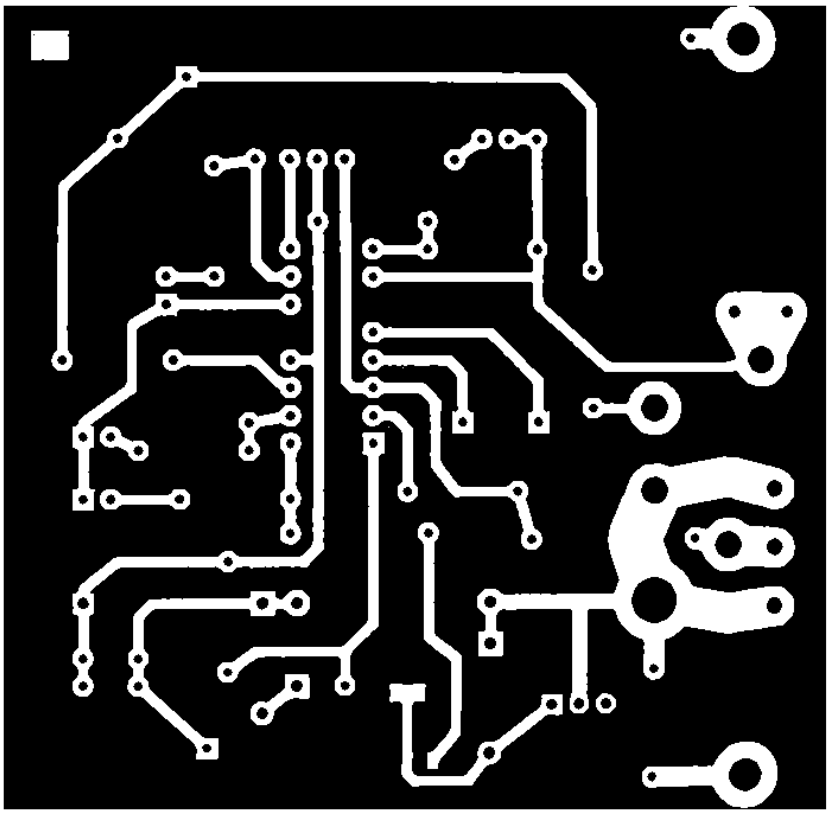
2.11 pav. Tikrinamos plokštės atvaizdas po pilno filtravimo

Atlikus atskirus takelio ir fono triukšmų filtravimus, gauname atvaizdą (2.11 pav.). Prieš ieškant takelių ir trumpiklių defektų, 2.11 paveikslas yra invertuojamas, nes ieškant nurodytų defektų tikrinami juodi pikseliai, baltų pikselių fone. Invertuotas 2.11 paveikslas pateiktas 2.12 paveiksle.

Iš etaloninio atvaizdo iškerpam visus objektus, kurie nėra maksimalaus ploto, t. y. pagrindas, tokiu būdu gaunami tikri etaloniniai takeliai (2.13 pav.).



2.12 pav. Tikrinamos plokštės invertuotas atvaizdas

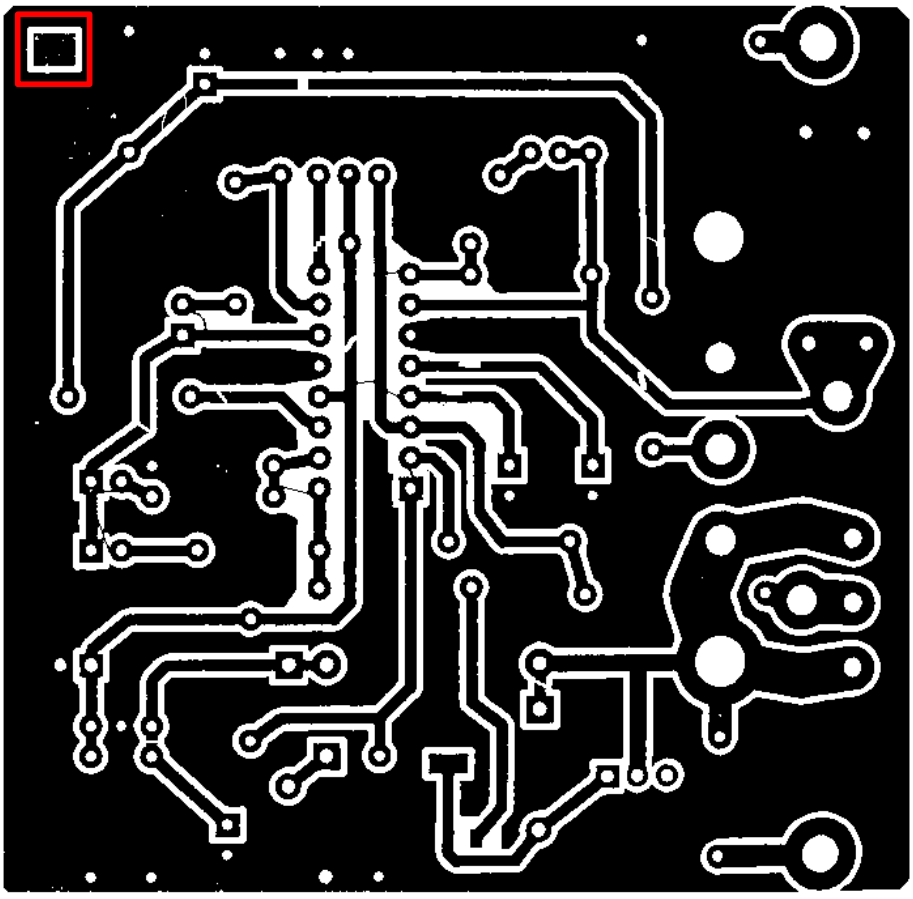


2.13 pav. Etaloninėje plokštėje išskirti objektai

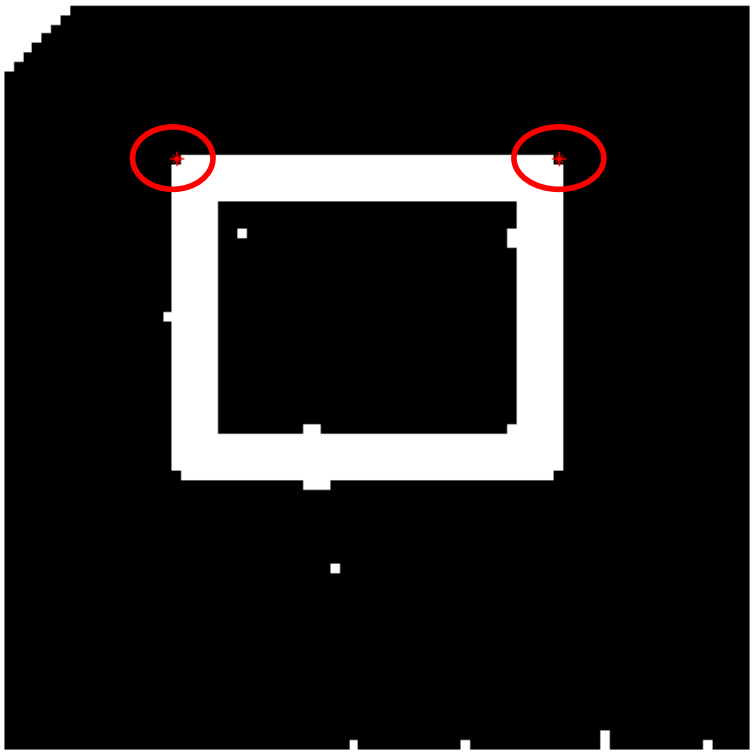
2.2.3 Tikrinamo atvaizdo lygiavimas etaloninio atvaizdo atžvilgiu

Dėl PCB atvaizdo fiksavimo įrenginių techninių vibracijų, gautas tikrinamos PCB plokštės atvaizdas gali būti šiek tiek pasisukęs. Jeigu tikrinamas atvaizdas, etaloninio atvaizdo atžvilgiu, bus šiek tiek pasikreipęs, po atlikto palyginimo gausime labai didelius neatitikimus, nes visi tikrinamo atvaizdo elementai yra pasukti į šoną.

Norint to išvengti, prieš atvaizdų lyginimą, reikia patikrinti tikinamo atvaizdo padėtį pagal numatytus taškus. Gautos tikrinamo atvaizdo numatytų taškų koordinatės yra palyginamos su su analogiškai gautomis etaloninio atvaizdo numatytų taškų koordinatėmis.

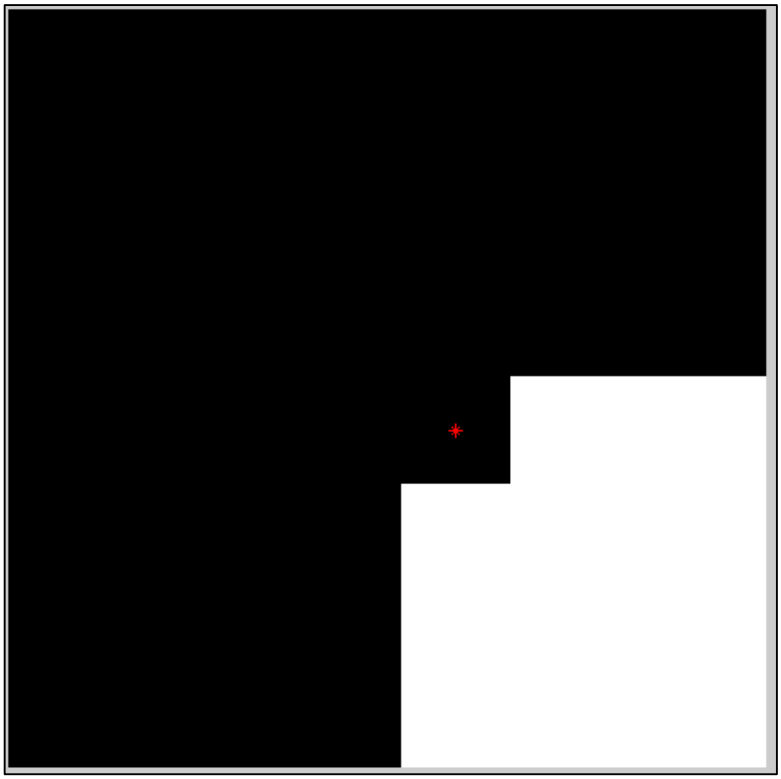


2.14 pav. Tikrinamo atvaizdo numatytų taškų vieta (apibraukta raudonai)



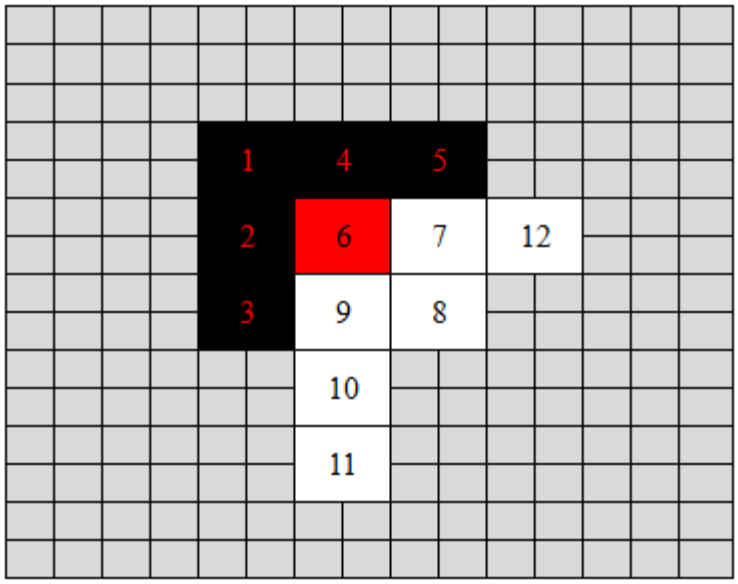
2.15 pav. Tikrinamo atvaizdo numatytų taškų vieta (apibraukta raudonai)

Kadangi tikrinamo atvaizdo nukrypimas dažniausiai būna nedidelis, tai atvaizdo numatyti taškai bus panašioje vietoje. Pagal užduotą sąlygą ieškomi du kampai (2.15 pav.). Norint išvengti ilgos paieškos, dviejų kampų paieškos zona yra sumažinama.



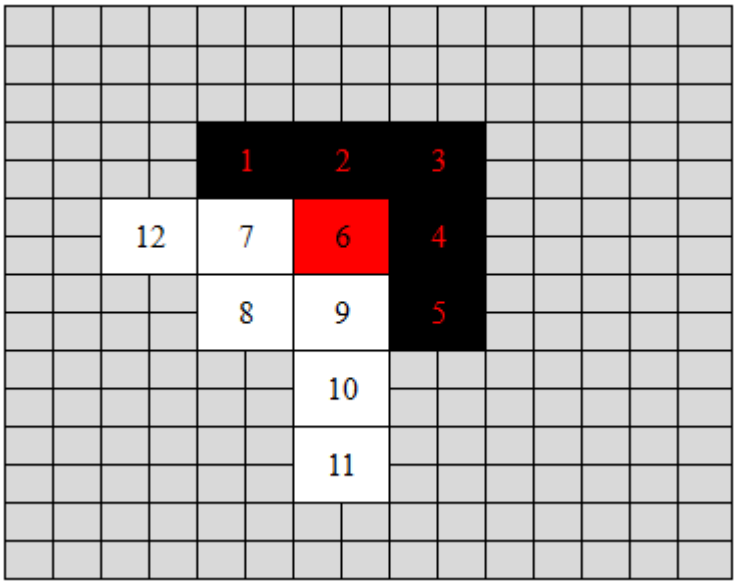
2.16 pav. Kairysis tikrinimo taškas

Ieškant kariojo tikrinimo taško tikrinama sąlyga, kurioje nurodyta, kad jeigu aplink juodą pikselį fiksuotomis koordinatėmis yra išsidėstę balti pikseliai ir dalis juodų pikselių. Jei randamos taško koordinatės, kurios tenkina sąlygą (2.17 pav.), jos yra įrašomos į masyvą.



2.17 pav. Kairiojo tikrinimo taško radimo kaukė

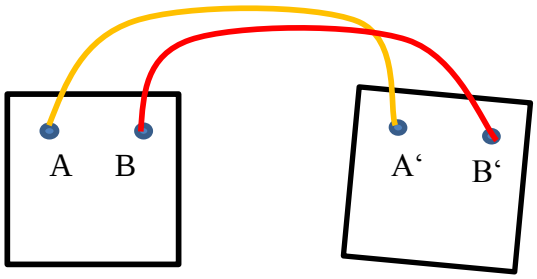
Kairiojo taško radimo sąlyga yra tokia: rastas taškas yra priskiriamas tikrinimui, jeigu (2.17 pav.) prie 6 numeriu pažymėto pikselio atitinkamose vietose (1, 2, 3, 4, 5) esantys pikseliai yra juodi ir (7, 8, 9, 10, 11, 12) vietose esantys pikseliai yra balti.



2.18 pav. Dešiniojo tikrinimo taško radimo kaukė

Dešiniojo kampo radimas yra analogiškas kaip ir kariojo kampo radimas, tačiau skiriasi tikrinamų baltų ir juodų pikselių koordinatės (2.18 pav.), kurios turi tenkinti užduotą salygą.

Kadangi etaloninis ir tikrinamas atvaizdai yra vienodo dydžio ir tikrinimo kampai yra toje pačioje plokštės vietoje, galima palyginti tikrinamos plokštės atvaizdo pasisukimą su etalonės plokštės atvaizdo pasisukimu.



2.19 pav. Neatitikimų tarp tikrinamų atvaizdų ieškojimas

Tikrinamiems taškams priskiriami pavadinimai. Etaloninio atvaizdo posūkio tikrinimo taškai įvardijami kaip A ir B, tikrinamo atvaizdo posūkio tikrinimo taškai atitinkamai įvardijami kaip A‘ ir B‘(2.19 pav.). Kai abiejų atvaizdų posūkio taškai yra rasti, užfiksuojamos tų taškų koordinatės. Taško A koordinatės pažymimos kaip x ir y, taško B koordinatės – x‘ ir y‘, taškoA‘ koordinatės – x^* ir y^*, taško B’ koordinatės – x^*‘ ir y^*‘.

Naudojantis afiniąją posūkio transformacija tikrinamo atvaizdo taškų koordinatės yra pakeičiamos į tokias, su kuriomis tikrinamas atvaizdas susivienodina su etaloniniu atvaizdu.
Plokštumos $\prod$ atvaizdavimas į save pačią vadinamas transoformacija: $f : \prod \to \prod$

Tarkime, kad taškas *A* atvaizduojamas į tašką *B = f(A)*. Tada taškas *B* vadinamas taško *A* vaizdu, o taškas *A* vadinamas taško *B* pirmvaizdžiu.[10]

Abipus vienareikšmė tiesinė plokštumo transformacija vadinama afiniaja, jei kiekvieno jos taško *A* vaizdo *B* koordinatės (x^*, y^*) išreiškiamos per pirmvaizdžio koordinates (*x*, *y*) tokiu būdu:

$$\begin{cases} x^* = a_1x + b_1y + c_1 \\ y^* = a_2x + b_2y + c_2 \end{cases} \text{ ir } \begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix} \neq 0 \qquad 2.3$$

$$A = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \quad B = \begin{bmatrix} x^* \\ y^* \\ 1 \end{bmatrix}, \quad T = \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ 0 & 0 & 1 \end{bmatrix} \qquad 2.4$$

Tada lygčių sistemą (2.3) galima užrašyti matriciniu pavidalu: B = TA. Matrica T vadinama afiniosios transformacijos *f* transformacijų matrica. Koeficientai a_1, a_2, b_1, b_2, c_1, c_2 gali įgyti ir nulines reikšmes. Pavyzdžiui, jei c_1=c_2=0, tada matricos įgyja paprastesnį pavidalą:

$$A = \begin{bmatrix} x \\ y \end{bmatrix}, \quad B = \begin{bmatrix} x^* \\ y^* \end{bmatrix}, \quad T = \begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \end{bmatrix} \qquad 2.5$$

Plokštumos ∏ pasisukimas kampu α apie koordinačių pradžią yra tiesinė transformacija. Ją atitinkanti transformacijų matrica R yra tokio pavidalo:

$$R = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \qquad 2.6$$

Iš lygčių sistemos B = TA ieškosime matricos T ir taip rasime α reikšmes.

2.2.4 Takelio defektų radimas

Naudojant 3x3 tikrinimo matricą, patikrinamas kiekvienas atvaizdo pikselis. Taigi ta pati tikrinimo matrica naudojama ir defektų ieškojimui takeliuose. Ieškant defektų takeliuose, po atvaizdų apdorojimo ir suvienodinimo, kiekvienam atskiram, sujungtam ne kampu elementui, esančiam atvaizde, yra priskiriama skaitinė reikšmė.

Kadangi tikrinamame atvaizde, takeliuose esantys defektai yra realizuoti nelygiagrečiai x ir y ašims (2.20 pav.), kiekvienas toks defektas susidaro iš n atskirų elementų, kuriuos reikia sujungti į vieną visumą.

1	0	0	0	0	0	0	0	1
1	0	0	0	0	0	1	1	1
1	0	0	0	0	0	1	0	1
1	0	0	0	0	1	0	0	1
1	0	0	1	1	0	0	0	1
1	0	0	1	0	0	0	0	1
1	1	1	0	0	0	0	0	1
1	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	1

2.20 pav. Takelio defektas atvaizduotas binariniu masyvu

Kaip parodytą 2.20 paveiksle, defektas yra „skersas“ takelio atžvilgiu. *Matlab* programavimo aplinkoje, pikseliai bus identifikuoti, kaip vienas elementas, jeigu jų reikšmė (1 arba 0) yra vienoda ir jie vienas su kitu jungsis tiesiogiai, ne kampu, naudojantis funkcija bwlabel.

1	0	0	0	0	0	0	0	1
1	0	0	0	0	0	1	1	1
1	0	0	0	0	0	1	0	1
1	0	0	0	0	1	0	0	1
1	0	0	1	1	0	0	0	1
1	0	0	1	0	0	0	0	1
1	1	1	0	0	0	0	0	1
1	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	1

2.21 pav. Takelio defekto pavienių pikselių grupavimas

Takelio defektas, pavaizduotas 2.21 paveiksle, bus sudarytas iš 4 atskirų pikselių grupių. Norint identifikuoti tokį takelio defektą, visas 4 pikselių grupes reikia sujungti.

1	0	0	0	0	0	0	0	1
1	0	0	0	0	0	5	5	1
1	0	0	0	0	0	5	0	1
1	0	0	0	0	4	0	0	1
1	0	0	3	3	0	0	0	1
1	0	0	3	0	0	0	0	1
1	2	2	0	0	0	0	0	1
1	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	1

2.22 pav. Sugrūpuoti takelio defekto pavieniai pikseliai

Jei 3x3 tikrinimo matrica nustato, kad pikselių grupė nr.2 jungiasi kampu su kita pikselių grupe, šiuo atveju su pikseliais pažymėtais 3 numeriu, visi trejetai yra perverčiami į dvejetus (2.23 pav.).

1	0	0	0	0	0	0	0	1
1	0	0	0	0	0	5	5	1
1	0	0	0	0	0	5	0	1
1	0	0	0	0	4	0	0	1
1	0	0	2	2	0	0	0	1
1	0	0	2	0	0	0	0	1
1	2	2	0	0	0	0	0	1
1	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	1

2.23 pav. Sugrūpuotos dvi pikselių grupės

Sugrūpavus dvejetais ir trejetais pažymėtus pikselius iš dviejų pikselių grupių gavome vieną, kurią pažymėjome dvejetais, tikrinimas tęsiamas toliau.

1	0	0	0	0	0	0	0	1
1	0	0	0	0	0	2	2	1
1	0	0	0	0	0	2	0	1
1	0	0	0	0	2	0	0	1
1	0	0	2	2	0	0	0	1
1	0	0	2	0	0	0	0	1
1	2	2	0	0	0	0	0	1
1	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	1

2.24 pav. Sugrūpuotos visos pikselių grupės susijungusios kampu

Atlikus atskirų pikselių grupių sujungimą išryškėja takelio įtrūkimas (2.24 pav.) kuris bus detektuojamas kaip defektas. Takelio įtrūkimo defektai surandami tikrinant kaimyninių pikselių reikšmę įtrūkimo vietoje.

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3	3	3	3	3	3	3	1	1	1	2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	1	1	2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	1	1	2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	1	1	2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	1	1	2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	1	1	1	2	2	2	2	2	2	2	2	2	2
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

2.25 pav. Pertraukto takelio grupavimas

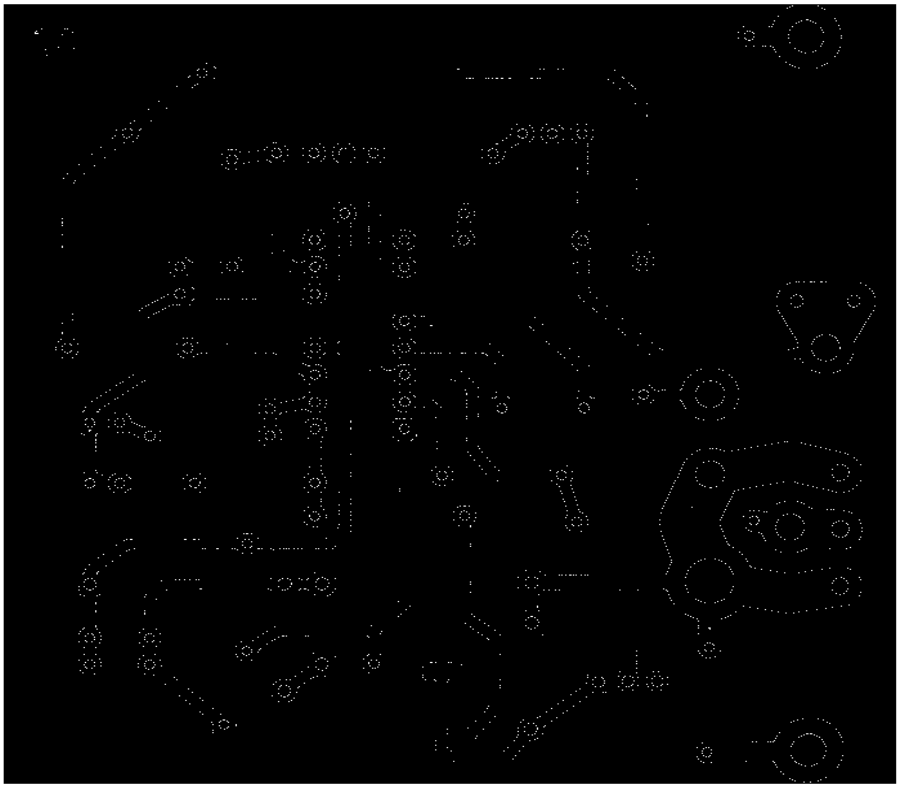
Tikrinimo metu, suradus įtrūkimą takelyje, tikrinamas kiekvienas pikselis įtrūkime. Jeigu surandamas skirtingos reikšmės pikselis, visam skirtingų pikselių masyvui suteikiama numeracija (2.25 pav.), mėlynai pažymėtiems pikseliams bus priskirtas žymėjimas trejetu, žalsvai pažymėtiems pikseliams bus priskirtas žymėjimas dvejetu. Po patikrinimo identifikuojamas takelio įtrūkimas. Takelio įtrūkimas detektuojamas tada, jei po patikrinimo rezultatai tenkina sąlygą, kuri aprašyta žemiau.

Mėlyną pikselių masyvą (2.25 pav), kuriam yra priskirtas trejetas, prilyginkime raidei „a“, žaslvą pikselių masyvą, kuriam priskirtas trejetas, prilyginkime raidei „b“.

Jeigu $a = 3$, o $b = 2$, tai reiškia, kad $a \neq 0$ ir $b \neq 0$, kas nusako, kad iš vientiso takelio pasidarė du atskiri takeliai.

2.26 pav. Detektuoti takelių įtrūkimai

Naudojant pikselių sugrupavimą realizuoti smulkūs takelio įtrūkimo defektai. Neišvengiamai takeliuose lieka „šiukšlių“, kurios priskiriamos prie toleruotinų defektų (2.27 pav.)



2.27 pav. Toleruotini takelių įtrūkimų defektai

2.2.5 Takelio dalinio defekto radimas

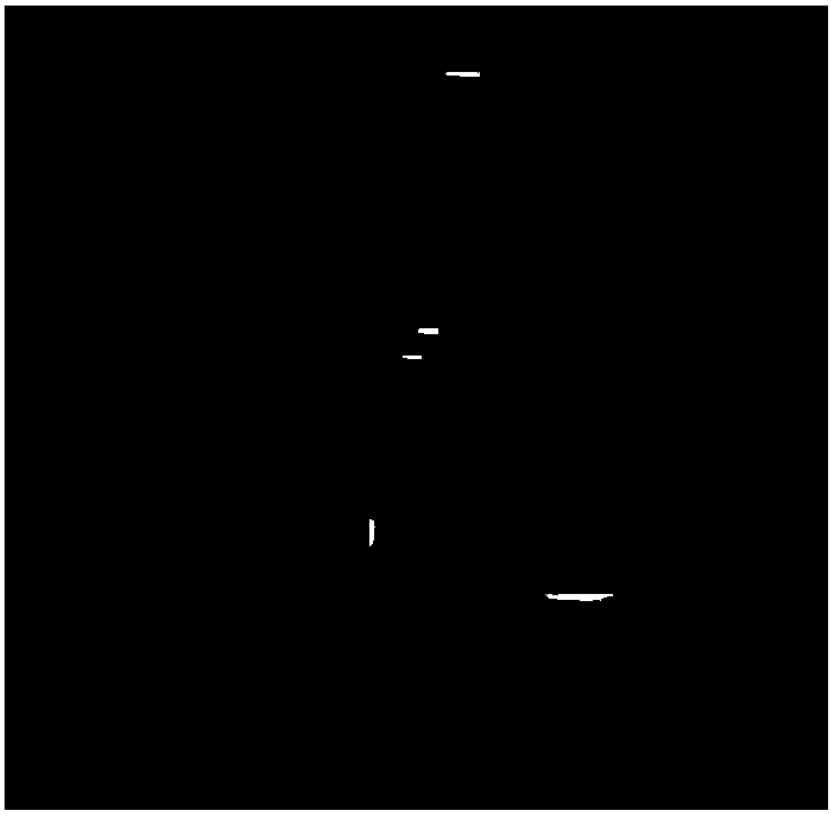
Takelio daliniam defektui rasti naudojama analogiška metodika, kaip ir ieškant takelio įtrūkimo defektų.

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3	3	3	3	3	3	3	1	1	1	1	3	3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3	1	1	1	3	3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3	1	1	1	3	3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

2.28 pav. Dalinio takelio defekto skaičių masyvas

Tikrinimo metu, suradus įtrūkimą takelyje, tikrinamas kiekvienas pikselis įtrūkime. Jeigu surandamas skirtingos reikšmės pikselis, visam skirtingų pikselių masyvui suteikiama numeracija (2.28 pav.), mėlynai pažymėtiems pikseliams bus priskirtas žymėjimas trejetu. Kadangi, patikrinus visus baltus pikselius, kurie yra įtrūkimo srityje, buvo rastas tik vienas skirtingos reikšmės pikselių masyvas, reiškia, kad defektas yra dalinis.

Jeigu a = 3, o b = 0, tai reiškia, kad a ≠ 0 ir b = 0, kas nusako, kad vientisame takelyje yra iškandimas.



2.29 pav. Daliniai takelio defektai

Rasti visi 5 daliniai takelių defektai atvaizduoti 2.29 paveiksle. Dėl defektų kūrimo, redaguojant etaloninį atvaizdą, susidarė papildomų „šiukšlių“, kurios priskiriamos prie toleruotinų defektų (2.30 pav.).

2.30 pav. Toleruotini daliniai takelio defektai

2.2.6 Trumpiklio defekto radimas

Trumpiklis sujungia du skirtingus takelius, todėl reikia tikrinti ar yra ryšys tarp takelių.

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3	3	3	3	3	3	3	1	1	1	1	1	4	4	4	4	4	4	4	4
3	3	3	3	3	3	3	1	1	1	1	1	4	4	4	4	4	4	4	4
3	3	3	3	3	3	3	1	1	1	0	0	4	4	4	4	4	4	4	4
3	3	3	3	3	3	3	1	1	0	1	1	4	4	4	4	4	4	4	4
3	3	3	3	3	3	3	1	0	1	1	1	4	4	4	4	4	4	4	4
3	3	3	3	3	3	3	0	1	1	1	1	4	4	4	4	4	4	4	4
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

2.31 pav. Trumpiklio defekto skaičių masyvas

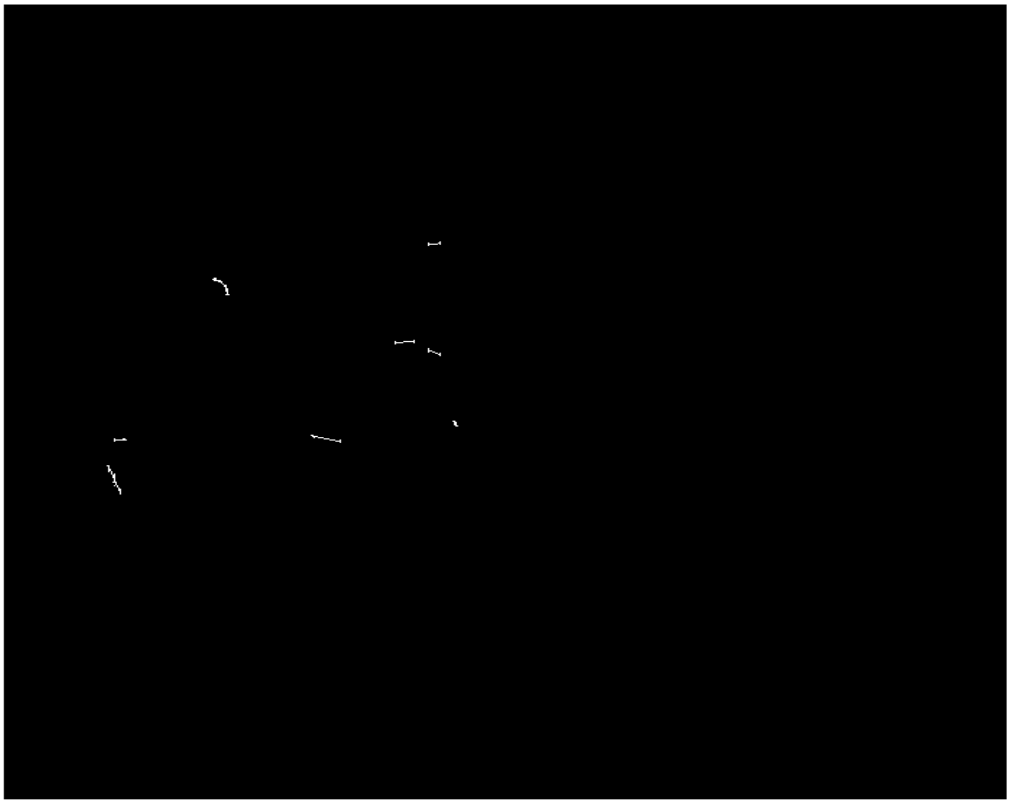
Trumpiklio defektai yra ieškomi baltoje srityje, tikrinant ar yra juodos spalvos pikselių ten kur jų nėra etaloniniame atvaizde. Baltoje srityje aptikus juodą pikselį, tikrinama jo kaiminystėje esančio pikselio reikšmė, o gretimam takeliui priskiriama skaitinė reiškmė (2.31 pav.). Jeigu atliekant tikrinimą, juodos spalvos pikseliai pasiekia kito takelio kraštą, tai tam takeliui taipogi priskiriama skaitinė reikšmė.

Jei a = 3, o b = 4, vadinasi, kad a ≠ b, kas nusako, kad sujungtų takelių priskirta skaitinė reikšmė yra nevienoda. Jeigu a ≠ b vadinasi, kad takeliai yra užtrumpinti.

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
1	1	1	1	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1
1	1	1	1	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

2.32 pav. Toleruotino trumpiklio defekto skaičių masyvas

Būna tokių atvejų, kad takelis užsitrumpina pats save, o tai nėra defektas. Jei tikrinant baltą sritį, bus rastas toks defektas, kuris atvaizduotas 2.32 paveiksle, tikrinimas vyks tol, kol bus patikrintos visų kaiminystėje esančių pikselių reikšmės. Jei a = 3, o b = 0 (nerastas), vadinasi, kad a = 3, kas nusako, kad baltoje srityje rasti juodi pikseliai nėra trumpiklio defektas.



2.33 pav. Rasti trumpiklio defektai

2.34 pav. Toleruotini trumpiklio defektai

2.2.7 Lydviečių defekto radimas

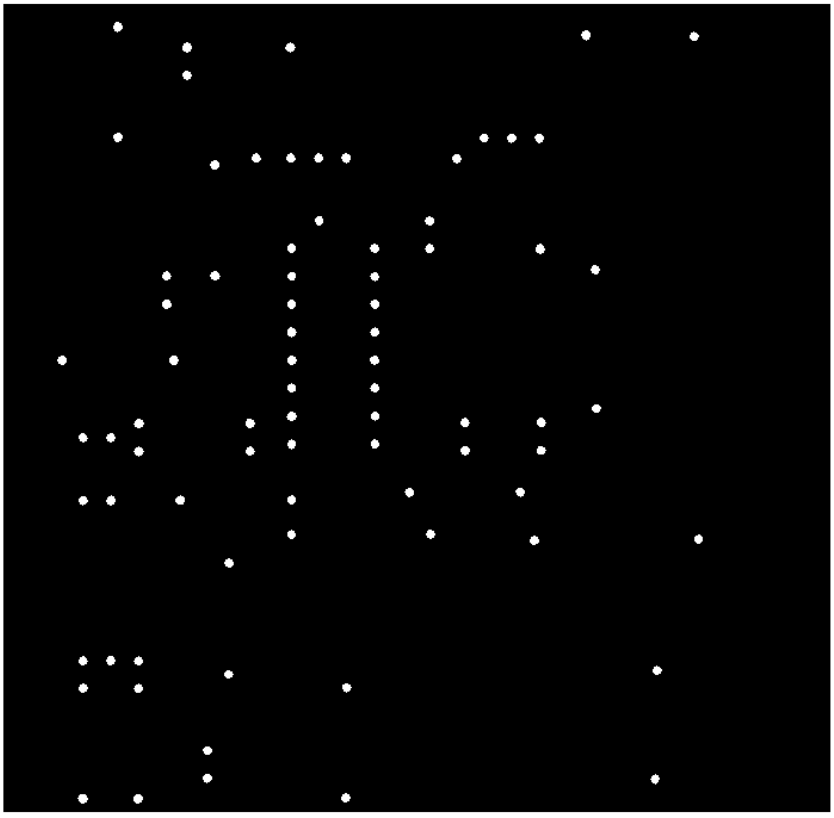
Norint rasti lydviečių defektus, reikia išsiskirti pačias lydvietes. Lydvietės išskiriamos etaloniame atvaizde. Naudodamasis *Matlab* pakete siūloma funkcija „r*egionprops*“, randami visi apskriti objektai. Kiekvienam apskritam objektui yra suskaičiuojamas perimetras ir plotas.

Apskaičiuojama objekto metrika: $$met = \frac{P^2}{4*\pi*S}$$ 2.7

P – objekto perimetras
S – objekto plotas

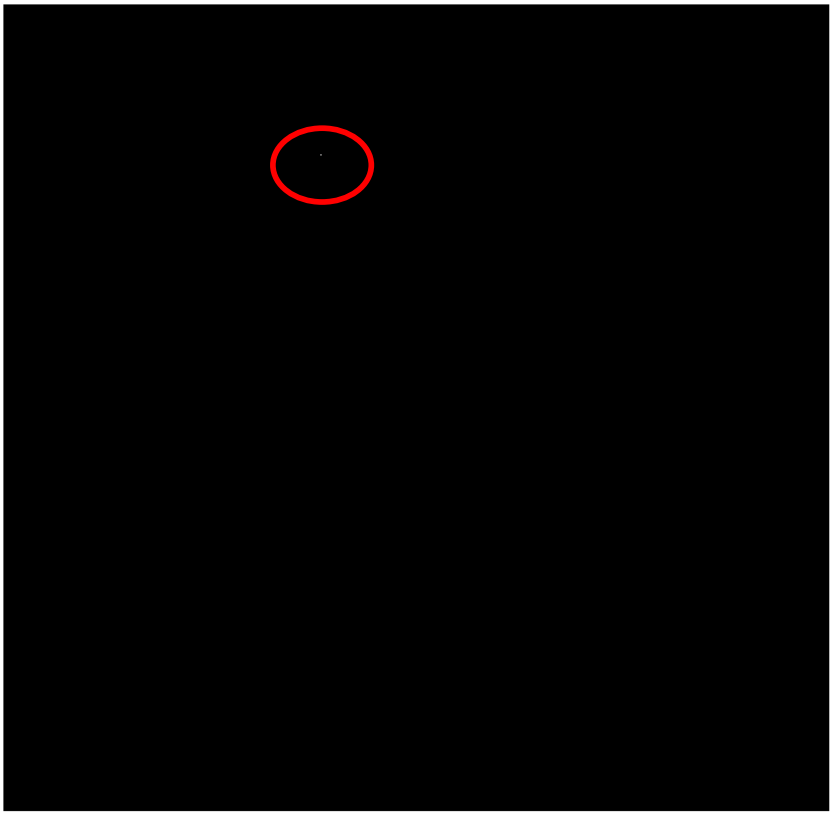
Jeigu metrika < 2, vadinasi objektas yra apskritimas.

Siekiant išvengti visokių didelių ertmių skirtų PCB tvirtinimui prie kokio nors pagrindo, naudojama apskritimų išskyrimo riba nuo 6,75 iki 8,25. Išskirtos lydvietės pavaizduotos 2.35 paveiksle. Defektai randami palyginus etaloninio atvaizdo lydviečių diametrus su tikrinamo atvaizdo lydviečių diametrais.



2.35 pav. Lydvietės išskirtos iš etaloninio atvaizdo

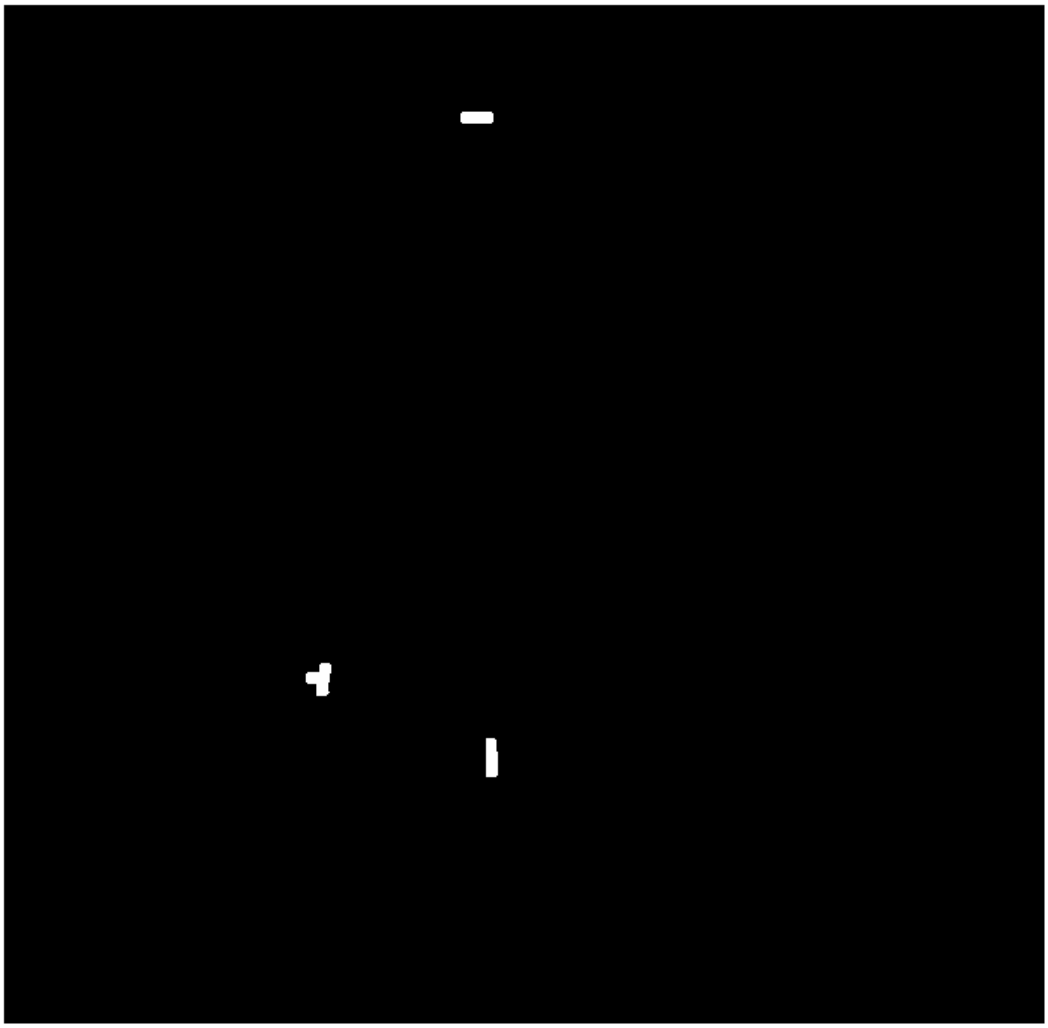
2.36 pav. Netoleruotini lydviečių defektai



2.37 pav. Toleruotinos lydviečių klaidos

2.2.8 Fono klaidų radimas

Fono klaidos randamos palyginus etaloninį ir tikrinamą atvaizdus. Iš etaloninio atvaizdo atėmus tikrinamą atvaizdą pasilieka tik neatitikimai (2.38 pav.).



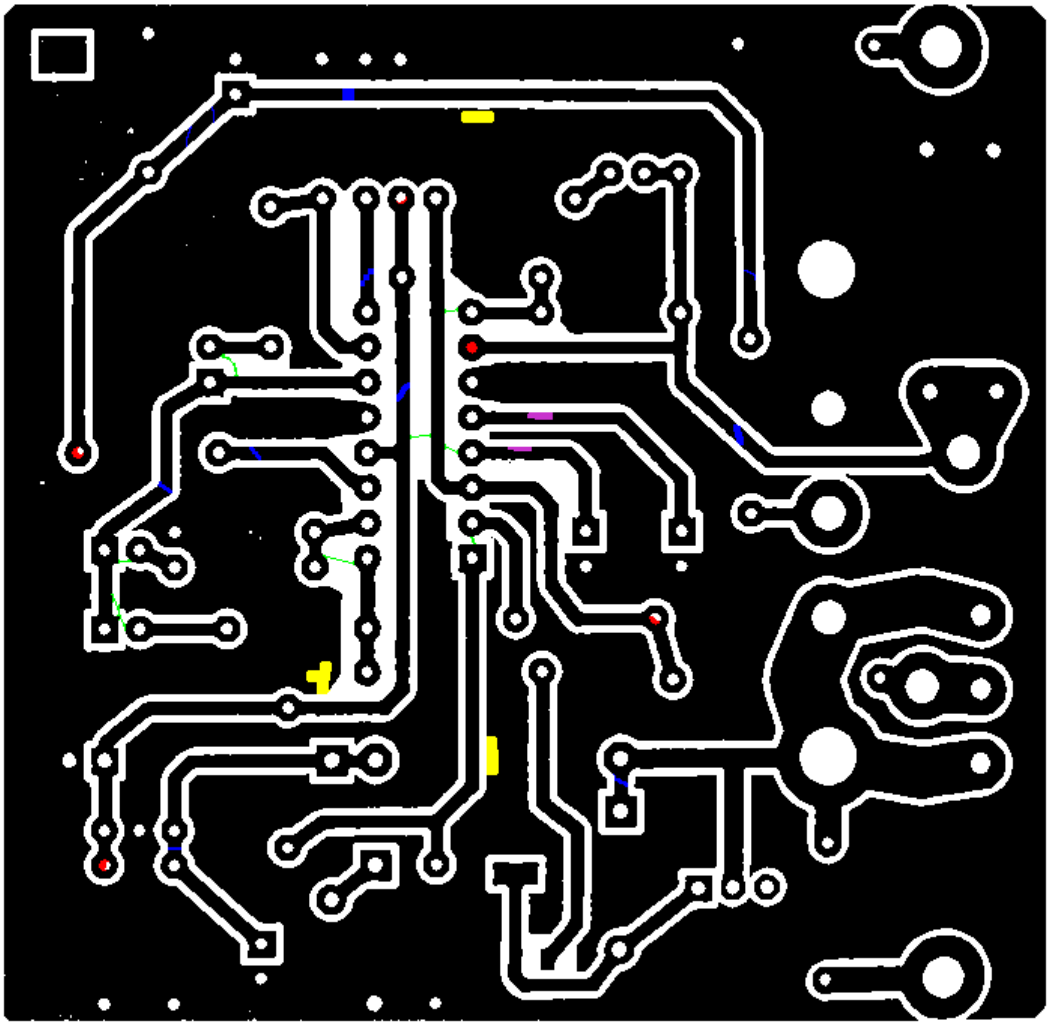
2.38 pav. Fono klaidos

2.2.9 Spalvinis klaidų žymėjimas ir jų registravimas

Radus kiekvieną defektą, jis pažymimas spalvine koduote ir yra atvaizduojamas tirkinamoje plokštėje (2.39 pav.).

2.1 lentelė. Defektų spalvinis žymėjimas

Defekto rūšis	Spalva
Trumpiklis	Žalia
Takelio trūkimas	Mėlyna
Lydvietės defektas	Raudona
Fono klaida	Geltona
Sienos klaida	Rausva



2.39 pav. Galutinis tikrinamos plokštės atvaizdas su sužymėtais defektais

Patikrinus visus defektus, juos atvaizdavus galutinėje formoje, visi defektai yra suskaičiuojami ir įrašomi į *excel* failą pavadinimu Ats.

2.1 lentelė. Defektų kiekio registravimas

Defekto pavadinimas	Kiekis
Takelių netoleruotinų klaidų skaičius	11
Takelių toleruotinų klaidų skaičius	2319
Trumpiklio netoleruotinų klaidų skaičius	8
Trumpiklio toleruotinų klaidų skaičius	6
Sienos netoleruotinų klaidų skaičius	2
Sienos toleruotinų klaidų skaičius	23
Lydviečių netoleruotinų klaidų skaičius	5
Lydviečių toleruotinu klaidų skaičius	1
Fono klaidų skaičius	3

3.TIRIAMOJI DALIS

Tiriamojoje dalyje ištirti atvaizdų filtravimo nuo triukšmų filtrai, kurie yra siūlomi *Matlab* programiniame pakete. PCB defektų inspektavimo metodo kūrimo metu buvo susidurta su problema, kad bandant išfiltruoti triukšmus su *Matlab* programiniame pakete siūlomais filtrais, kartu su triukšmais buvo išfiltruoti ir dalis plokštėje esančių defektų, kurie buvo ganėtinai maži (~1 pikselio pločio).

Tyrimo metu, buvo ištirti tokie *Matlab* programiniame pakete siūlomi filtrai :

1. Netiesiniai filtrai: medianinis filtras, standartinio nuokrypio filtras
2. Morfologinės funkcijos : BWareaopen, clean, erode

Po kiekvieno filtro ir morfologinės valymo funkcijos išbandymo rezultatai bus apsprendžiami iš suskaičiuotų defektų galutiniame rezultate.

Palyginimui, etaloninės defektų detektavimo skaitinės reikšmės imamos iš metodinės dalies galutinio rezultato defektų kiekio registro (3.1 lentelė).

3.1 lentelė. Etaloninės defektų detektavimo skaitinės reikšmės

Defekto pavadinimas	Kiekis
Takelių netoleruotinų klaidų skaičius	11
Takelių toleruotinų klaidų skaičius	2319
Trumpiklio netoleruotinų klaidų skaičius	8
Trumpiklio toleruotinų klaidų skaičius	6
Sienos netoleruotinų klaidų skaičius	2
Sienos toleruotinų klaidų skaičius	23
Lydviečių netoleruotinų klaidų skaičius	5
Lydviečių toleruotinu klaidų skaičius	1
Fono klaidų skaičius	3

3.1 Medianinis filtras

Šis filtras naudoja lokalius metodus, kaip ir tiesiniai filtrai, tačiau jis nenaudoja sąsūkos (teisingiaum sąsūkos ir neįmanoma panaudoti mediano filtro išraiškai). Filtras randa filtruoto pikselio intensyvumą, imdamas aplink pikselį esančių kaimynų intensyvumo medianą. Medianos operatorius dažniausiai būna implementuojamas ieškant medianos 3 x 3 dydžio pikselių plote, tačiau medianos galima ieškoti ir didesnio ploto zonose. Taip pat galima naidoti ne kvadrato formos zonas medianai rasti, pvz., galima imti medianą, esančią toje pačioje tiesėje vertikaliai arba horizontaliai kaip ir pikselis, kuriam priskiriame filtruotą reikšmę.[8]

Medianos filtras gerai tinka „*salt and peper*“tipo triukšmo pašalinti. „*Salt and peper*“ yra vadinamas triukšmas, kai vaizde atsiranda izoliuoti balti bei juodi pikseliai.Vienas iš medianos filtro pranašumų yra tai, kad filtruojant nedingsta kontūrai, o triukšmas yra sumažinamas.[7]

Pagrindiniai medianos filtro trūkumai: nors medianos filtras išsaugo kontūrus, tačiau juo yra pašalinami vaizdo minimumai ir maksimumai. Pasirinkus per didelio ploto filtrą, gali būti pašalinami kai kurie vaizde esantys objektai. [9]

Algoritmas**,** realizuojantis medianos filtrą, gali būti aprašytas tokiais žingsniais (A yra filtro kaukės dydis bei K filtravimų skaičius):

1. Išskiria taško kaukę, kurios dydis yra (A*A) ir kurios centras yra analizuojamame taške p_i;
2. Kaukės elementai išrūšiuojami didėjimo (gali būti ir mažėjimo) tvarka;
3. Išrenkamas centrinis išrūšiuotos kaukės elementas, ir jis įrašomas į taško p_i vietą;
4. Grįžtame į punktą 1. ir taikome jį taškui p_{i+1}

Neapibrėžtos algoritmo veikimo situacijos:

1. Filtro kaukės dalis lieka už paveikslėlio ribų
2. Filtro kaukės dydis yra lyginis skaičius, to pasekoje atliekant ciklo dalį neįmanoma išrinkti centrinio kaukės elemento.

Neapibrėžtų situacijų sprendimo būdai:

1. Imant paveikslėlio kontūrus, filtro kaukės dalis, likusi už paveikslėlio kontūrų, užpildoma nuliais;

2. Kaukės dydžiui esant lyginiam, išrenkant centrinį elementą iš išrūšiuotos kaukės elementų, imami du centriniai išrūšiuotos kaukės taškai, ir imamas jų aritmetinis vidurkis $(a_j+a_{j+1})/2$.

Algoritmo veikimo greitis, matuojamas operacijų skaičiumi:

$$T = K * BPP * (A^2)log(A^2) * N * M \quad 3.1$$

T – vidutinis algoritmo operacijų skaičius,

K – filtro pritaikymo paveisklėliui skaičius,

BPP – (BitsPerPixel) paveikslėlio bitų kiekis, skirtas vienam taškeliui išsaugoti,

A – matricos dydis; $(A^2)log(A^2)$ – matricos rūšiavimo algoritmo greitis,

N – paveikslėlio aukštis (taškų skaičius),

M – paveikslėlio plotis (taškų skaičius).

Panaudojus medianinį filtrą gauti tokie rezultatai:

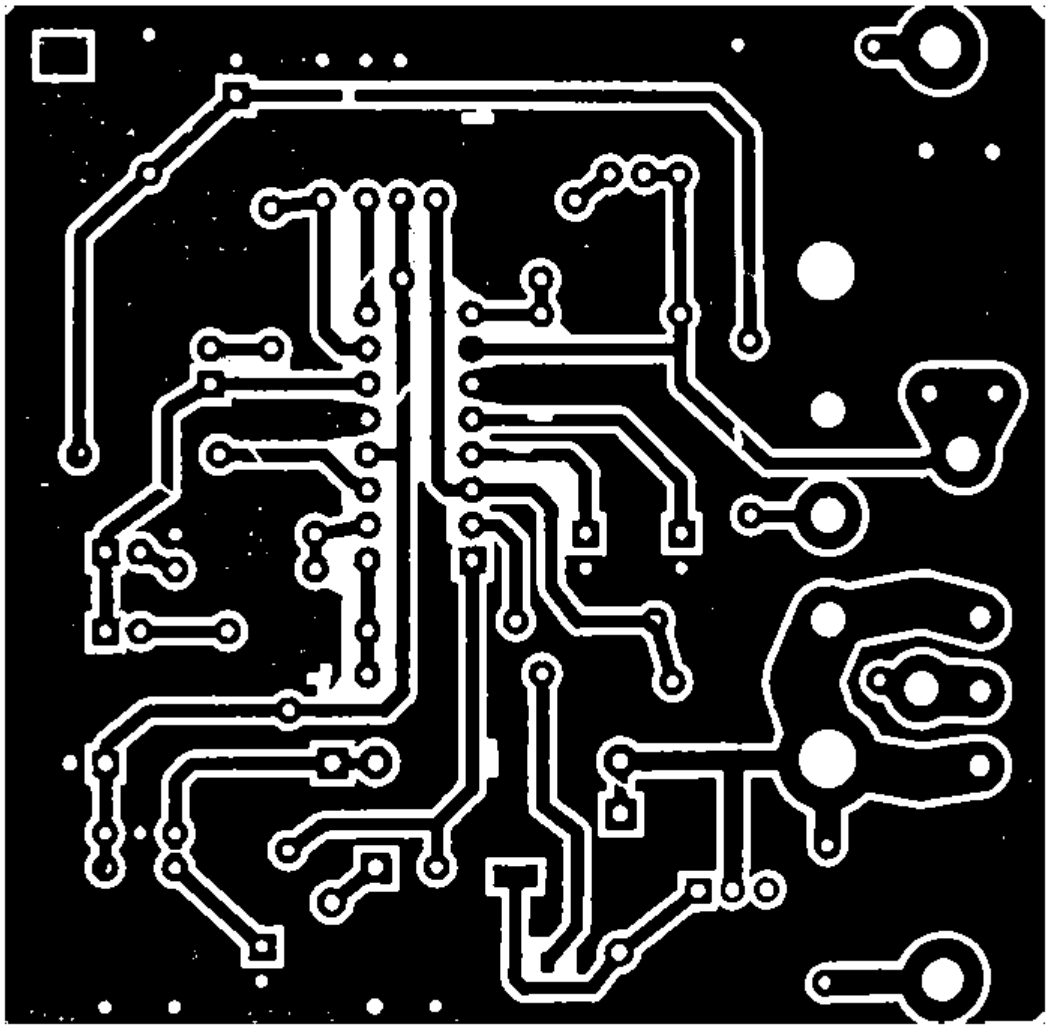
3.2 lentelė. Defektų detektavimo skaitinės reikšmės panaudojus medianinį filtrą

Defekto pavadinimas	Kiekis
Takelių netoleruotinų klaidų skaičius	7
Takelių toleruotinų klaidų skaičius	2022
Trumpiklio netoleruotinų klaidų skaičius	0
Trumpiklio toleruotinų klaidų skaičius	31
Sienos netoleruotinų klaidų skaičius	4
Sienos toleruotinų klaidų skaičius	41
Lydviečių netoleruotinų klaidų skaičius	5
Lydviečių toleruotinu klaidų skaičius	0
Fono klaidų skaičius	26

3.3 lentelė. Defektų detektavimo skaitinės reikšmės skirtumas lygininat su etaloninėmis reikšmėmis panaudojus medianinį filtrą

Defekto pavadinimas	Kiekis
Takelių netoleruotinų klaidų skaičius	-4
Takelių toleruotinų klaidų skaičius	-297
Trumpiklio netoleruotinų klaidų skaičius	-8
Trumpiklio toleruotinų klaidų skaičius	+25
Sienos netoleruotinų klaidų skaičius	+2
Sienos toleruotinų klaidų skaičius	+18
Lydviečių netoleruotinų klaidų skaičius	0
Lydviečių toleruotinu klaidų skaičius	-1
Fono klaidų skaičius	+23

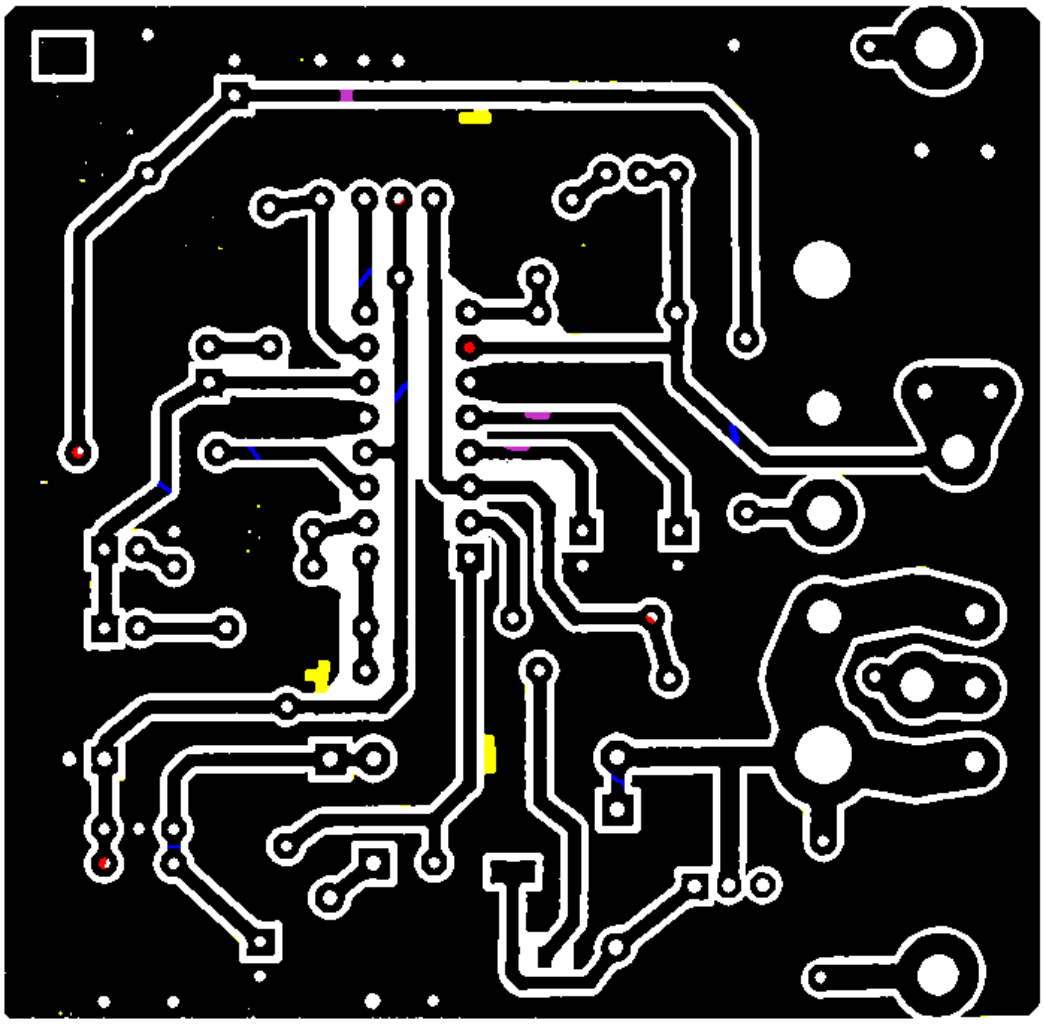
Po medianinio filtro panaudojimo, gautų skaitinių reikšmių skirtumas žymimas su minuso ženklu, jeigu naudojant šį triukšmo šalinimo filtrą gauti rezultatai, lyginant su etalonėmis skaitinėmis reikšmėmis, yra mažesni t. y., neatitikimų buvo rasta mažiau nei nurodo etaloninės skaitinės reikšmės. Jeigu skaitinių reikšmių skirtumas žymimas su pliuso ženklu, vadinasi neatitikimų padaugėjo.



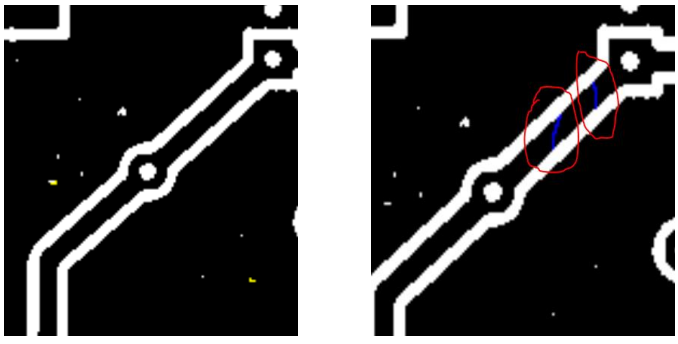
3.1pav. Tikrinamos PCB atvaizdas po medianinio filtro panaudojimo

Kaip matome iš 3.5 paveikslo, kartu su triukšmu buvo išfiltruoti ir visi trumpikliai, dalis smulkių įtrūkimų takeliuose (3.3 pav). Didesni takelio įtrūkimai buvo detektuoti kaip sienos klaida (3.3 pav.). Padaugėjo fono klaidų (3.6 pav.).

Medianinio filtro naudojimas, šiuo atveju, netenkina siekiamų uždavinio sprendimo rezultatų. Šis filtras gali būti naudojamas tada, kai triukšmas filtruojamas atvaizduose, kuriems nebus taikomi itin tikslias reikšmes nustatantys algoritmai.



3.2 pav. Ištestuotas PCB atvaizdas po medianinio filtro panaudojimo

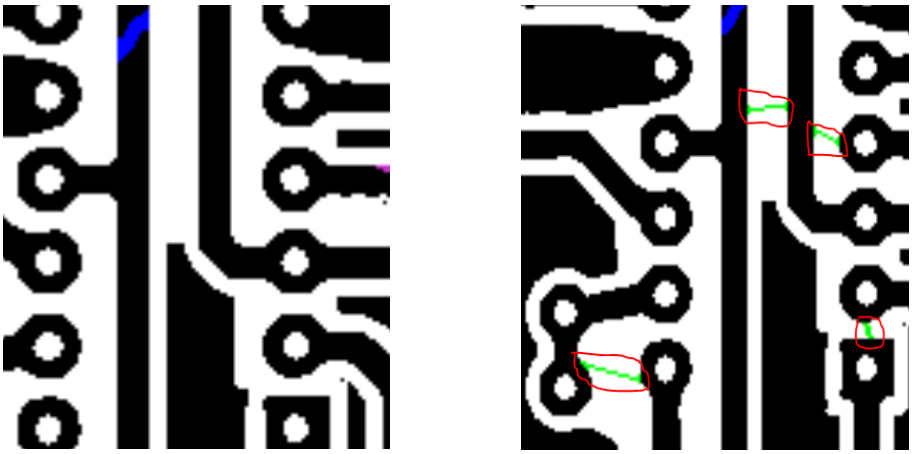


(a) (b)

3.3 pav. Išfiltruoti takelių įtrūkimai a), takelių įtrūkimai etaloniniame atvaizde b)

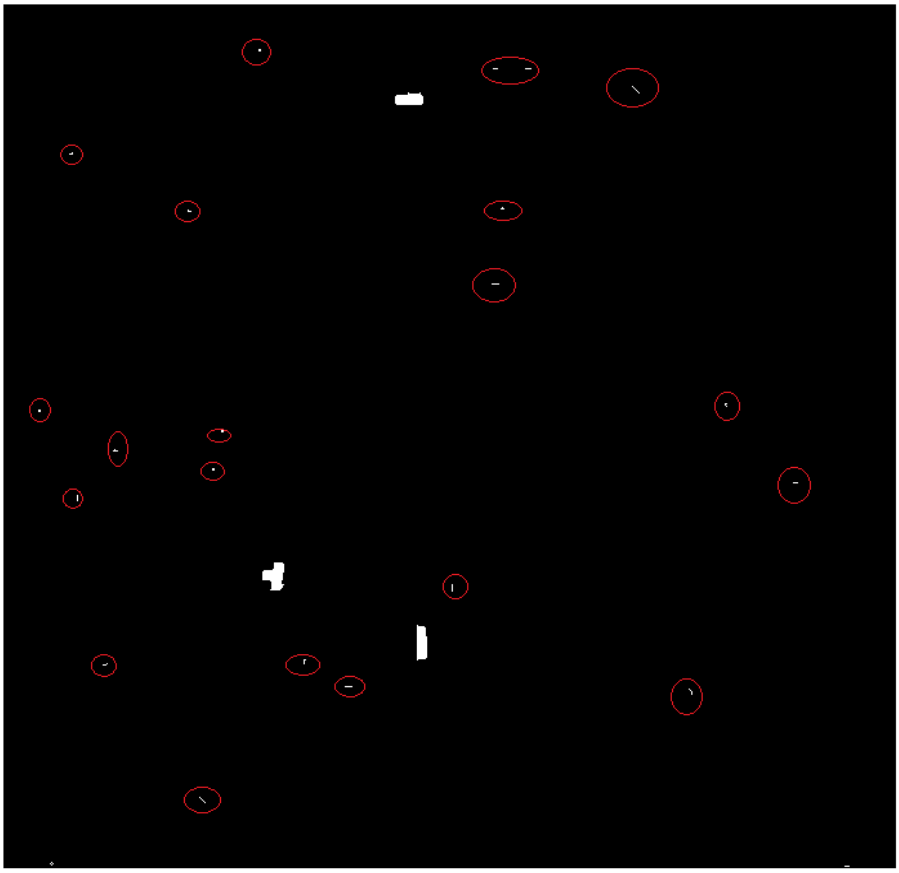
(a) (b)

3.4 pav. Takelio įtrūkimas detektuotas kaip sienos klaida a), takelio įtrūkimas etaloniame atvaizde b)

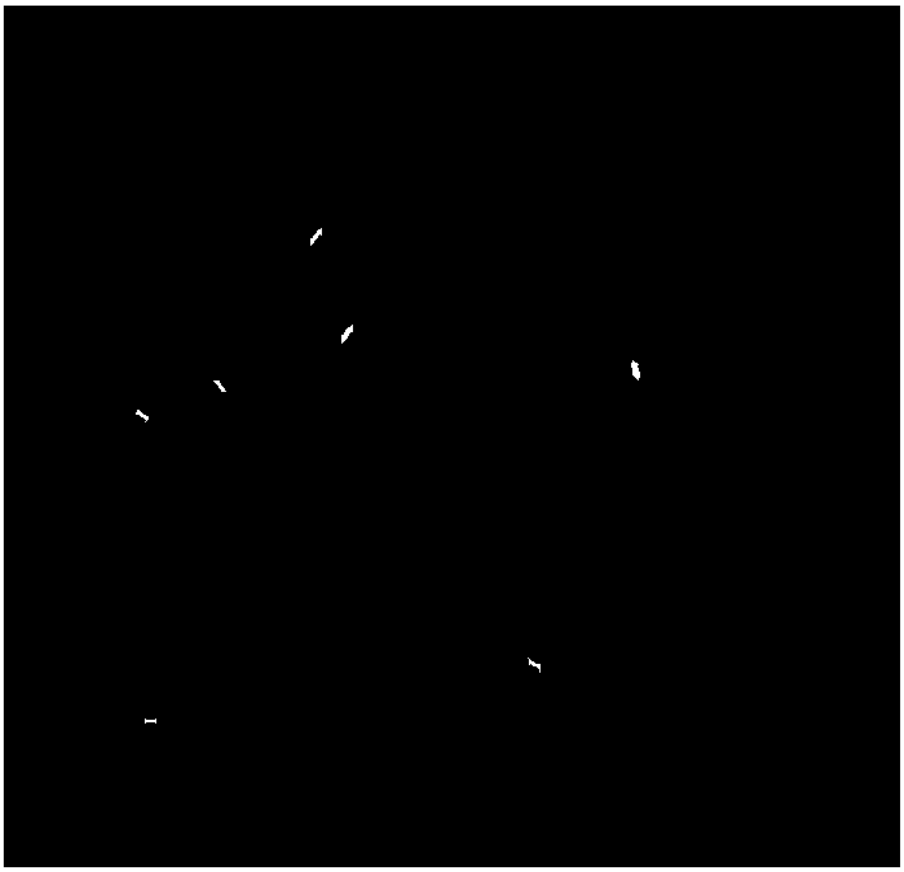


(a) (b)

3.5 pav. Išfiltruoti visi trumpikliai a), trumpikliai etaloniame atvaizde b)



3.6 pav. Papildomai atsiradusios fono klaidos po medianinio filtro panaudojimo



3.7 pav. Po medianinio filtravimo likę takelio defektai

3.2 Standartinio nuokrypio filtras

Naudojant standartinio nuokrypio filtrą tikrinama 3x3 matricos, viduriniojo pikselio reikšmė, gauta reikšmė yra prilyginama aplinkinių pikselių reikšmėms.[2]

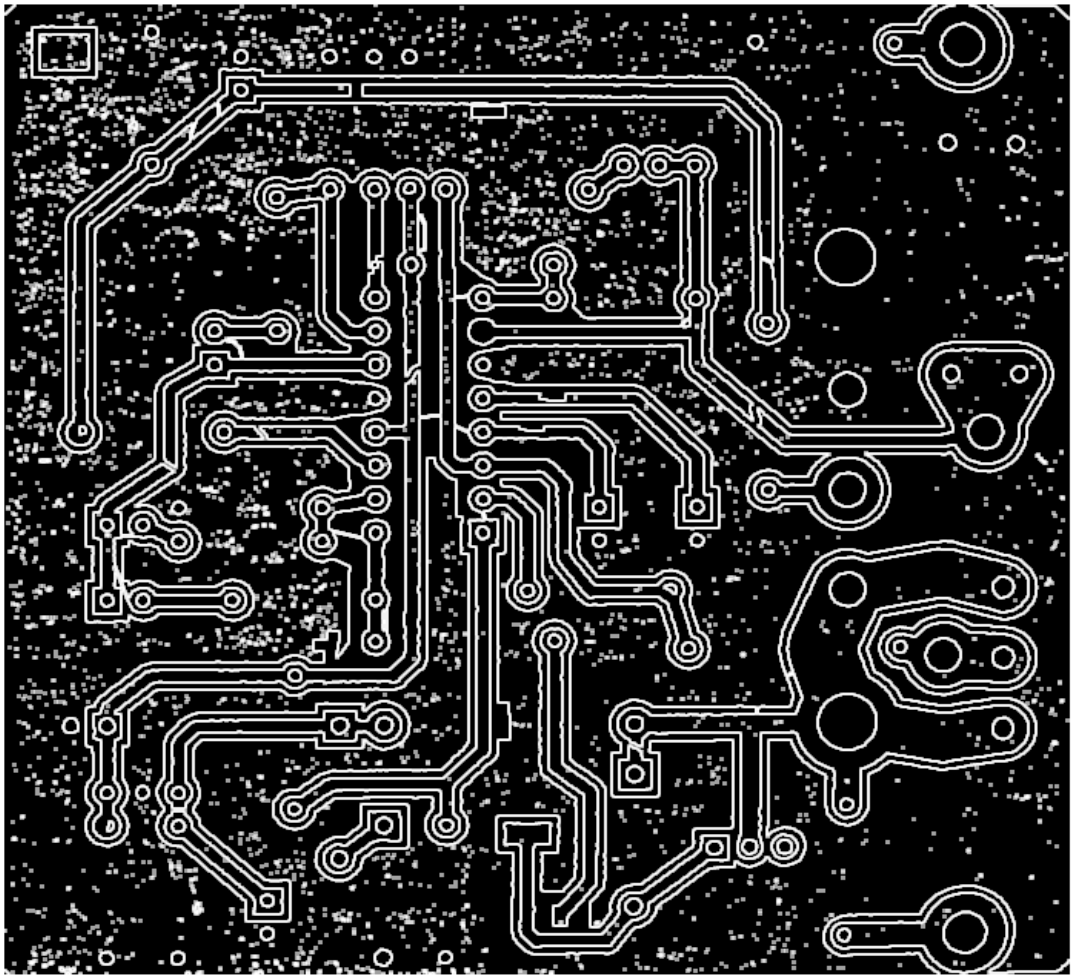
3.4 lentelė. Defektų detektavimo skaitinės reikšmės panaudojus standartinio nuokrypio filtrą

Defekto pavadinimas	Kiekis
Takelių netoleruotinų klaidų skaičius	0
Takelių toleruotinų klaidų skaičius	94
Trumpiklio netoleruotinų klaidų skaičius	2
Trumpiklio toleruotinų klaidų skaičius	10
Sienos netoleruotinų klaidų skaičius	120
Sienos toleruotinų klaidų skaičius	0
Lydviečių netoleruotinų klaidų skaičius	78
Lydviečių toleruotinu klaidų skaičius	0
Fono klaidų skaičius	1881

3.5 lentelė. Defektų detektavimo skaitinės reikšmės skirtumas lyginant su etaloninėmis reikšmėmis panaudojus standartinio nuokrypio filtrą

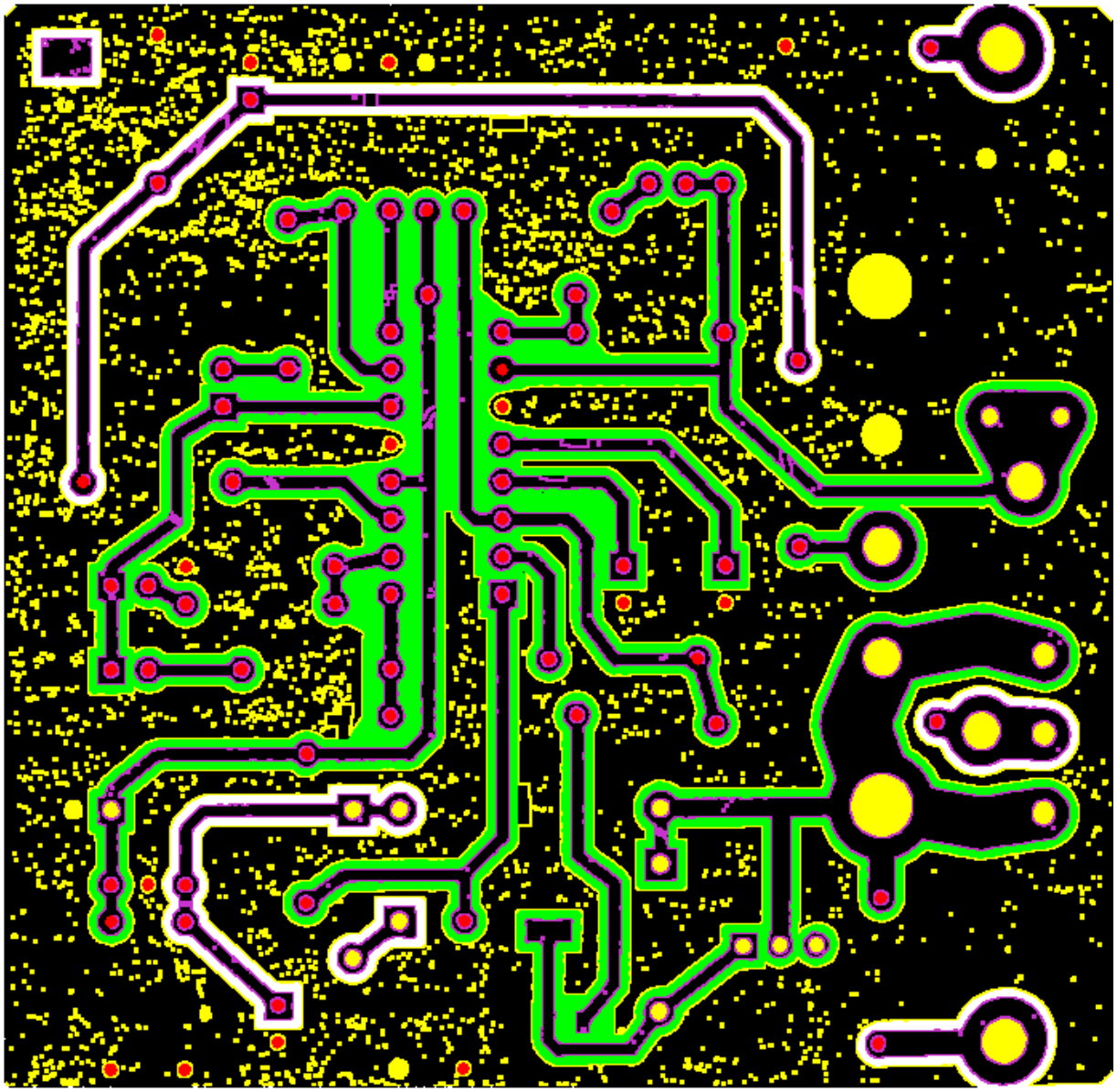
Defekto pavadinimas	Kiekis
Takelių netoleruotinų klaidų skaičius	-12
Takelių toleruotinų klaidų skaičius	-2225
Trumpiklio netoleruotinų klaidų skaičius	-6
Trumpiklio toleruotinų klaidų skaičius	+4
Sienos netoleruotinų klaidų skaičius	+118
Sienos toleruotinų klaidų skaičius	-23
Lydviečių netoleruotinų klaidų skaičius	+73
Lydviečių toleruotinu klaidų skaičius	-1
Fono klaidų skaičius	+1878

Po standartinio nuokrypio filtro panaudojimo, gautų skaitinių reikšmių skirtumas žymimas su minuso ženklu, jeigu naudojant šį triukšmo šalinimo filtrą gauti rezultatai, lyginant su etalonėmis skaitinėmis reikšmėmis yra mažesni t. y., neatitikimų buvo rasta mažiau nei nurodo etaloninės skaitinės reikšmės. Jeigu skaitinių reikšmių skirtumas žymimas su pliuso ženklu, vadinasi neatitikimų padaugėjo.



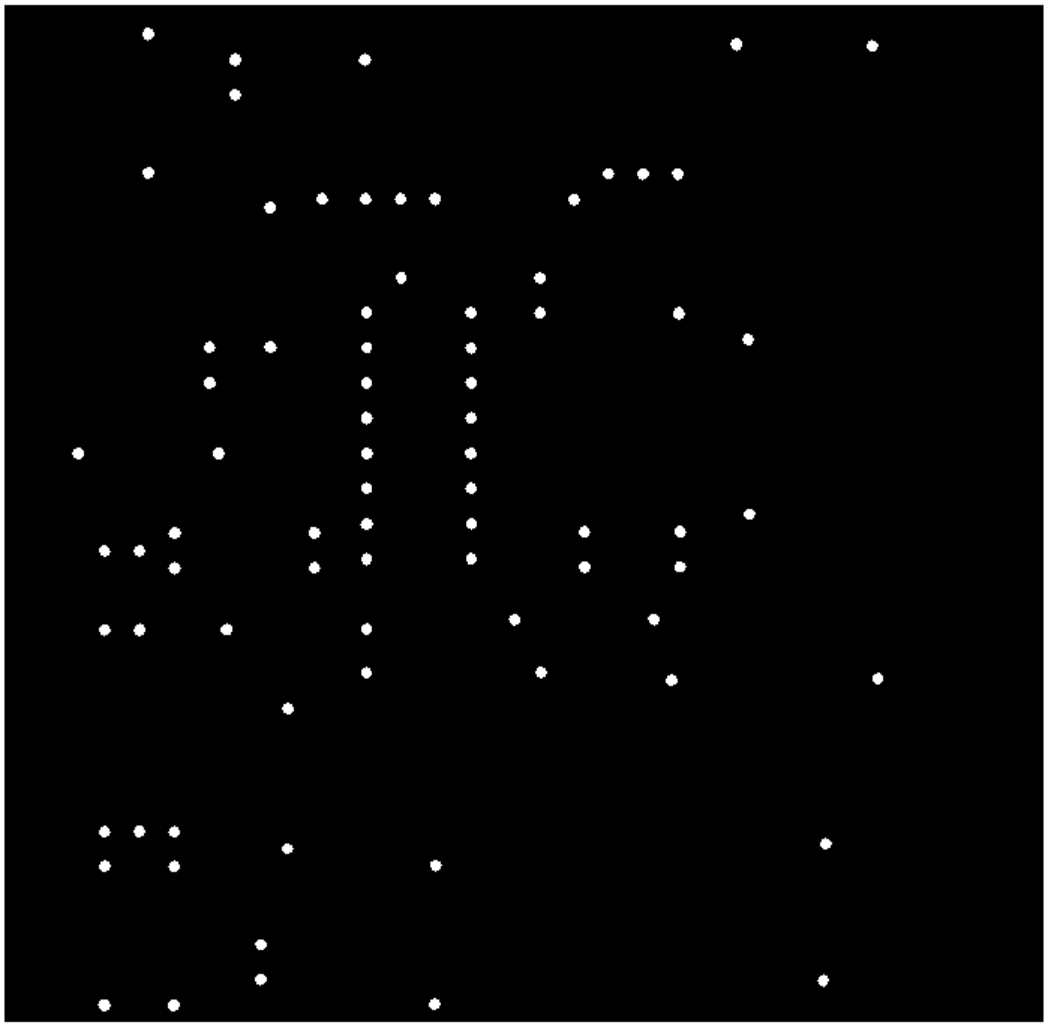
3.8 pav. Tikrinamos PCB atvaizdas po standartinio nuokrypio filtro panaudojimo

Naudojant standartinio nuokrypio filtrą, tikrinamame atvaizde yra „priauginama“ dar daugiau triukšmingų pikselių grupių darinių (3.8 pav.), ko pasekoje tikslus PCB defektų detektavimas nėra galimas.



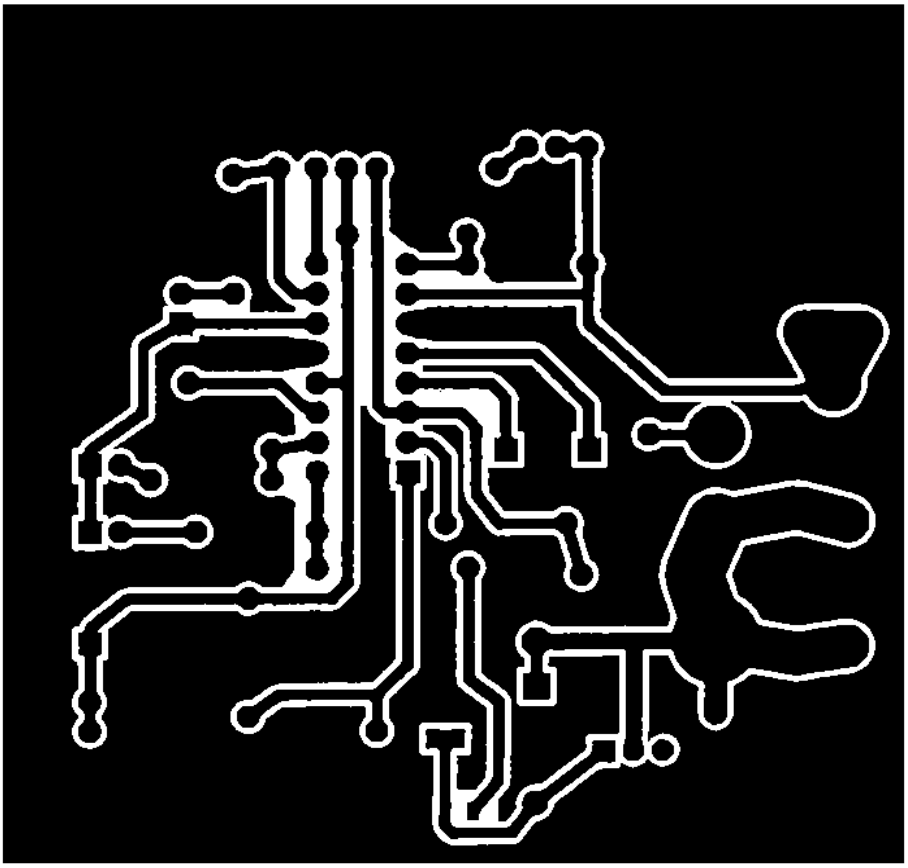
3.9 pav. Ištestuotas PCB atvaizdas po standartinio nuokrypio filtro panaudojimo

Atlikus visas tikrinamo atvaizdo apdorojimo ir testavimo operacijas, gautas galutinis ištestuotos PCB atvaizdas visiškai netenkina iekiamų uždavinio sprendimo rezultatų.

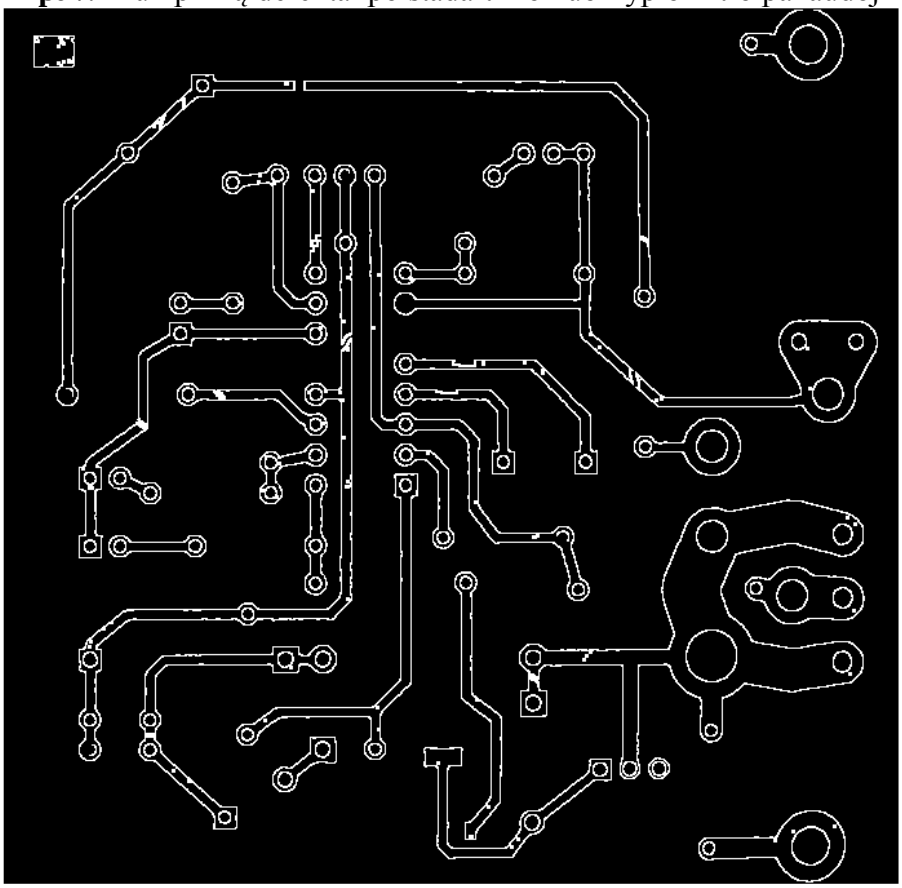


3.10 pav. Lydviečių defektai po stadartinio nuokrypio filtro panaudojimo

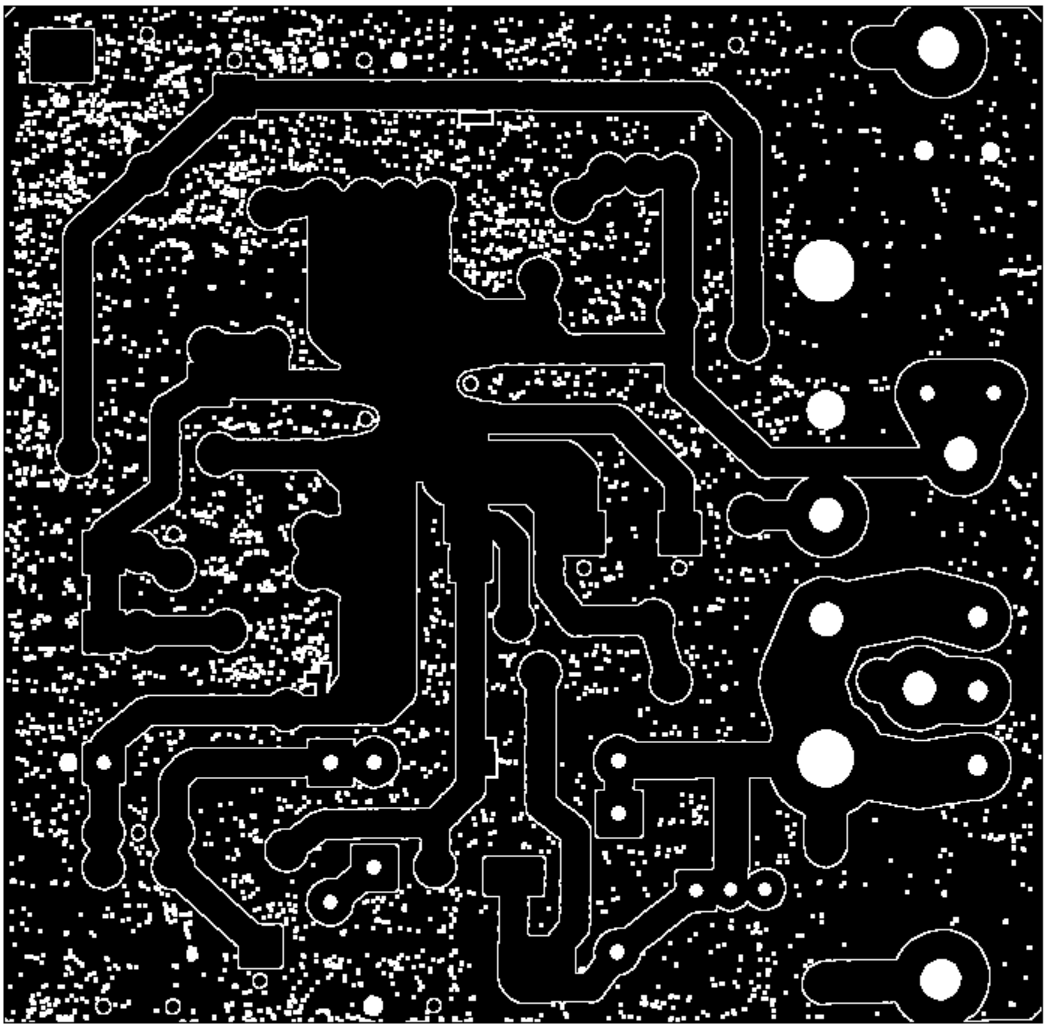
Panaudojus standartinių nuokrypių filtrą, buvo išfiltruotos visos lydvietės (3.10 pav.), ko pasekoje lyginant su etaloniniu atvaizdu, visos lydvietės buvo detektuotos kaip defektai.



3.11 pav. Trumpiklių defektai po stadartinio nuokrypio filtro panaudojimo



3.12 pav. Sienos klaidos po stadartinio nuokrypio filtro panaudojimo



3.13 pav. Fono klaidos po stadartinio nuokrypio filtro panaudojimo

3.3 Morfologinės operacijos „Bwareaopen“ naudojimas šalinant triukšmus

Morfologinė operacija „*Bwareaopen“*naudojama norint pašalinti visus mažus pikselių junginius binariniame atvaizde. Standartiškai parašyta funkcija pašalina visus mažesnius pikselių junginius, kurių bendra pikselių suma yra < 8 sujungtus pikselius. Galima koreguoti pikselių barjerą, todėl atvaizdas ištirtas su 4 ir su 8 sujungtų pikselių riba. [3]

3.3.1 Bwareaopen su 4 pikselių barjeru

Atliktas tikrinamo atvaizdo apdorojimas, naudojantis 4 pikselių barjeru. Gauti rezultatai pateikti 3.6, 3.7 lentelėse.

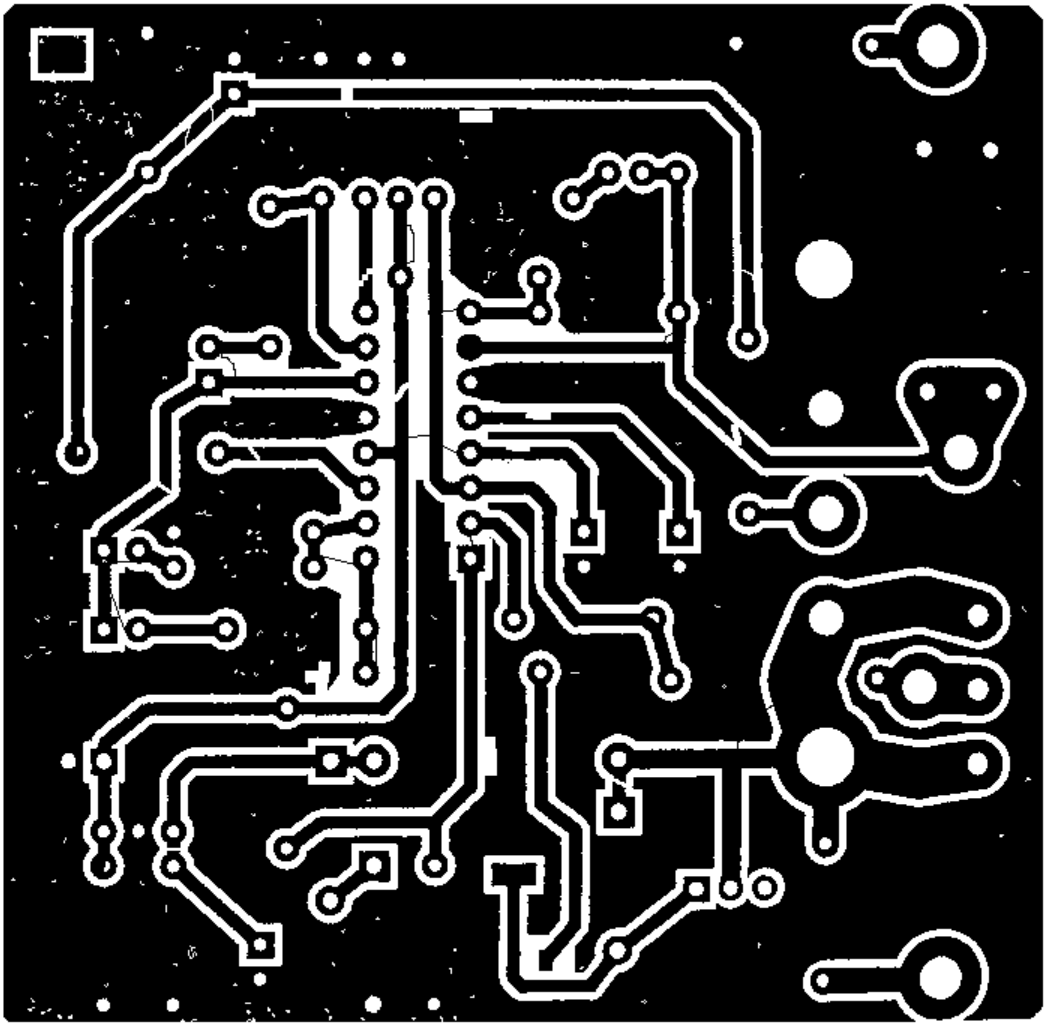
3.6 lentelė. Defektų detektavimo skaitinės reikšmės, panaudojus morfologinę opraciją „*Bwareaopen“* su 4 pikselių barjeru

Defekto pavadinimas	Kiekis
Takelių netoleruotinų klaidų skaičius	11
Takelių toleruotinų klaidų skaičius	2319
Trumpiklio netoleruotinų klaidų skaičius	8
Trumpiklio toleruotinų klaidų skaičius	5
Sienos netoleruotinų klaidų skaičius	2
Sienos toleruotinų klaidų skaičius	23
Lydviečių netoleruotinų klaidų skaičius	5
Lydviečių toleruotinu klaidų skaičius	0
Fono klaidų skaičius	182

3.7 lentelė. Defektų detektavimo skaitinės reikšmės skirtumas lyginant su etaloninėmis reikšmėmis panaudojus morfologinę opraciją „*Bwareaopen“* su 4 pikselių barjeru

Defekto pavadinimas	Kiekis
Takelių netoleruotinų klaidų skaičius	0
Takelių toleruotinų klaidų skaičius	0
Trumpiklio netoleruotinų klaidų skaičius	0
Trumpiklio toleruotinų klaidų skaičius	-1
Sienos netoleruotinų klaidų skaičius	0
Sienos toleruotinų klaidų skaičius	0
Lydviečių netoleruotinų klaidų skaičius	0
Lydviečių toleruotinu klaidų skaičius	-1
Fono klaidų skaičius	+179

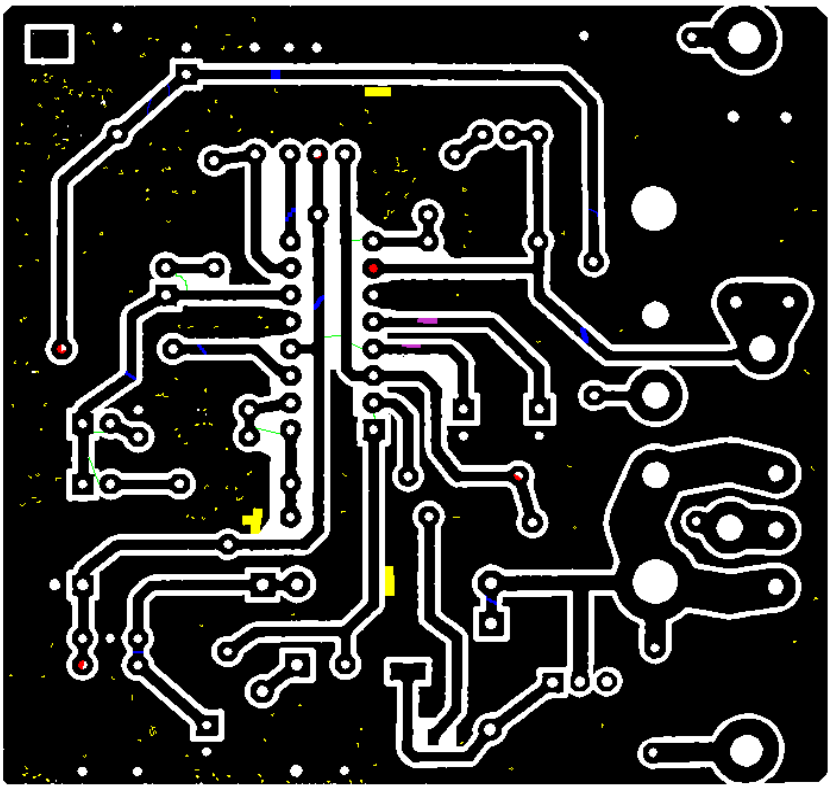
Po morfologinės operacijos „*Bwareaopen“* panaudojimo, gautų skaitinių reikšmių skirtumas žymimas su minuso ženklu, jeigu naudojant šią funkciją gauti rezultatai, lyginant su etalonėmis skaitinėmis reikšmėmis, yra mažesni t. y., neatitikimų buvo rasta mažiau nei nurodo etaloninės skaitinės reikšmės. Jeigu skaitinių reikšmių skirtumas žymimas su pliuso ženklu, vadinasi neatitikimų padaugėjo. Sutapus defektų skaitinėms reikšmėms, žymima 0.



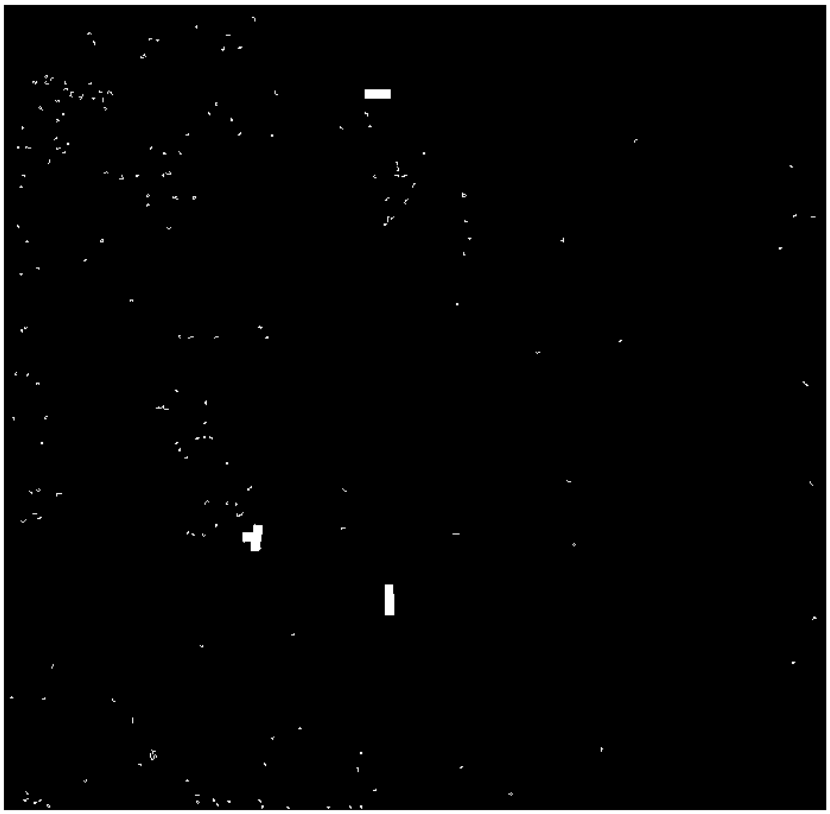
3.14 pav. Tikrinamos PCB atvaizdas po morfologinės operacijos „*Bwareaopen“* su 4 pikselių barjeru panaudojimo

Panaudojus „*Bwareaopen“* operaciją, išvalyta didelė dalis triukšmų (3.14 pav.). Po pavienių pikselių junginių, kurie yra mažesni nei 4 pikselių junginys, valymo matome, kad nebuvo pašalinti takelių smulkūs defektai kurių plotis ~1 pikselį.

Po valymo, lyginant su etalonėmis skaitinėmis reikšmėmis, padidėjo fono klaidų skaičius, kadangi dauguma pikselių junginių yra didesni, nei sujungti 4 pikseliai.



3.14 pav. Ištestuotas PCB atvaizdas po morfologinės operacijos „*Bwareaopen“* su 4 pikselių barjeru panaudojimo



3.15 pav. Neišvalytos fono klaidos po morfologinės operacijos „*Bwareaopen“* su 4 pikselių barjeru panaudojimo

3.3.2 Bwareaopen su 8 pikselių barjeru

Atliktas tikrinamo atvaizdo apdorojimas, naudojantis 8 pikselių barjeru. Gauti rezultatai pateikti 3.8, 3.9 lentelėse.

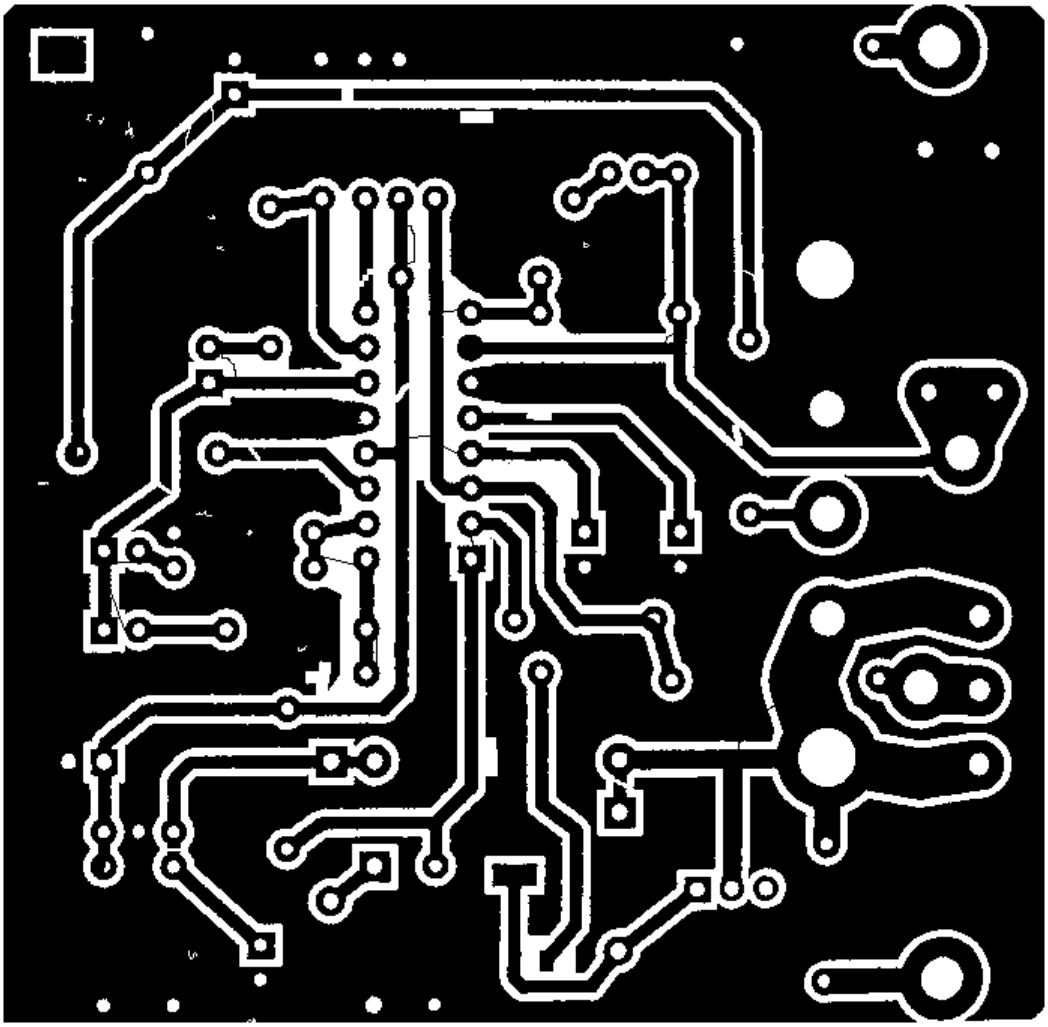
3.8 lentelė. Defektų detektavimo skaitinės reikšmės, panaudojus morfologinę opraciją „*Bwareaopen"* su 8 pikselių barjeru

Defekto pavadinimas	Kiekis
Takelių netoleruotinų klaidų skaičius	11
Takelių toleruotinų klaidų skaičius	2317
Trumpiklio netoleruotinų klaidų skaičius	8
Trumpiklio toleruotinų klaidų skaičius	5
Sienos netoleruotinų klaidų skaičius	2
Sienos toleruotinų klaidų skaičius	22
Lydviečių netoleruotinų klaidų skaičius	5
Lydviečių toleruotinu klaidų skaičius	0
Fono klaidų skaičius	15

3.9 lentelė. Defektų detektavimo skaitinės reikšmės skirtumas lyginant su etaloninėmis reikšmėmis panaudojus morfologinę opraciją „*Bwareaopen"* su 8 pikselių barjeru

Defekto pavadinimas	Kiekis
Takelių netoleruotinų klaidų skaičius	0
Takelių toleruotinų klaidų skaičius	-2
Trumpiklio netoleruotinų klaidų skaičius	0
Trumpiklio toleruotinų klaidų skaičius	-1
Sienos netoleruotinų klaidų skaičius	0
Sienos toleruotinų klaidų skaičius	-1
Lydviečių netoleruotinų klaidų skaičius	0
Lydviečių toleruotinu klaidų skaičius	-1
Fono klaidų skaičius	+12

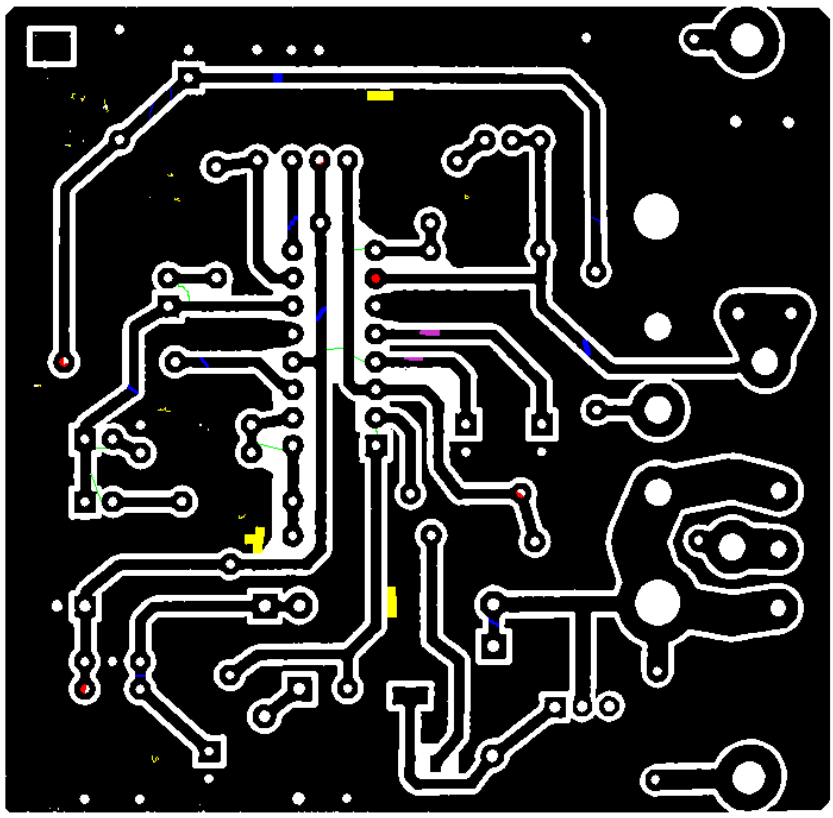
Po morfologinės operacijos „*Bwareaopen"* panaudojimo, gautų skaitinių reikšmių skirtumas žymimas su minuso ženklu, jeigu naudojant šią funkciją gauti rezultatai, lyginant su etalonėmis skaitinėmis reikšmėmis, yra mažesni t. y., neatitikimų buvo rasta mažiau nei nurodo etaloninės skaitinės reikšmės. Jeigu skaitinių reikšmių skirtumas žymimas su pliuso ženklu, vadinasi neatitikimų padaugėjo. Sutapus defektų skaitinėms reikšmėms, žymima 0.



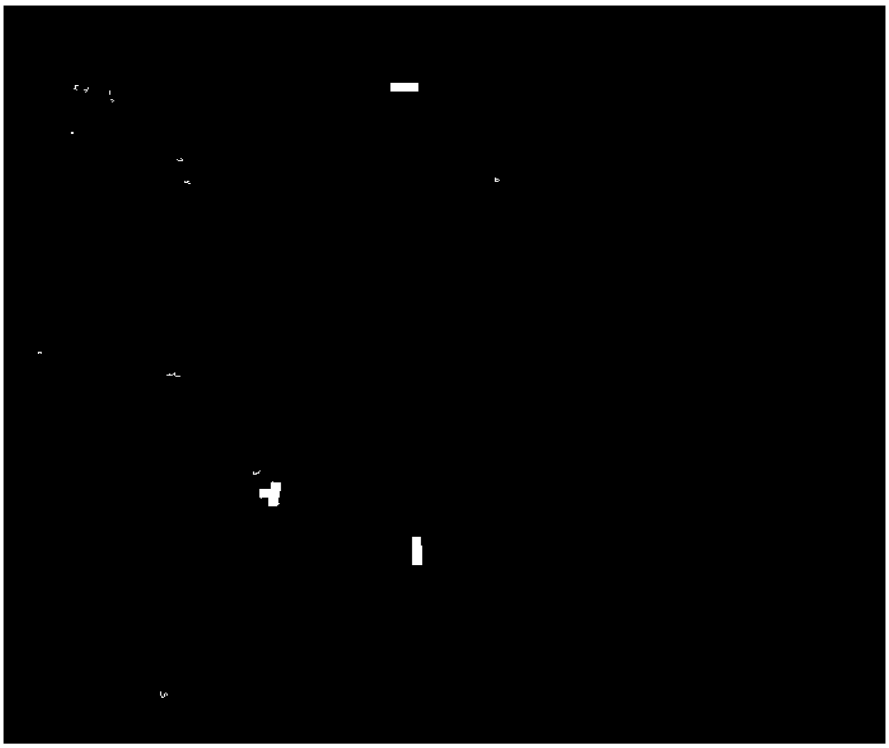
3.16 pav. Tikrinamos PCB atvaizdas po morfologinės operacijos „*Bwareaopen“* su 8 pikselių barjeru panaudojimo

Panaudojus „*Bwareaopen“* operaciją, išvalyta didelė dalis triukšmų (3.16 pav.). Po pavienių pikselių junginių, kurie yra mažesni nei 8 pikselių junginys, valymo matome, kad nebuvo pašalinti takelių smulkūs defektai kurių plotis ~1 pikselį.

Po valymo, lyginant su etalonėmis skaitinėmis reikšmėmis, padidėjo fono klaidų skaičius, kadangi likusių pikselių junginiai yra didesni, nei sujungti 8 pikseliai, tačiau lyginant su valymu, atliktu naudojantis operacija „*Bwareaopen“* su 4 pikselių barjeru, fono klaidų kiekis žymiai sumažėjo. Papildomai po valymo, lyginant su etaloninėmis skaitinėmis reikšmėmis, sumažėjo takelių, trumpiklių, sienos ir lydviečių toleruotinos klaidos.



3.17 pav. Ištestuotas PCB atvaizdas po morfologinės operacijos „*Bwareaopen“*su 8 pikselių barjeru panaudojimo



3.18 pav. Neišvalytos fono klaidos po morfologinės operacijos „*Bwareaopen“* su 8 pikselių barjeru panaudojimo

3.4 Morfologinės operacijos „erode“ naudojimas šalinant triukšmus

Binariniams vaizdmas apdoroti skirta morfologinė operacija erozija (angl. *erode*) apibrėžiama taip:

- Erozija : $X \ominus B = \{x: B_x \in X\}$,

Čia X – binarinis objektas, sudarytas iš taškų $\{x\}$, B – struktūrinis elementas, o B_x – struktūrinis elementas su centru taške *x*. [7]

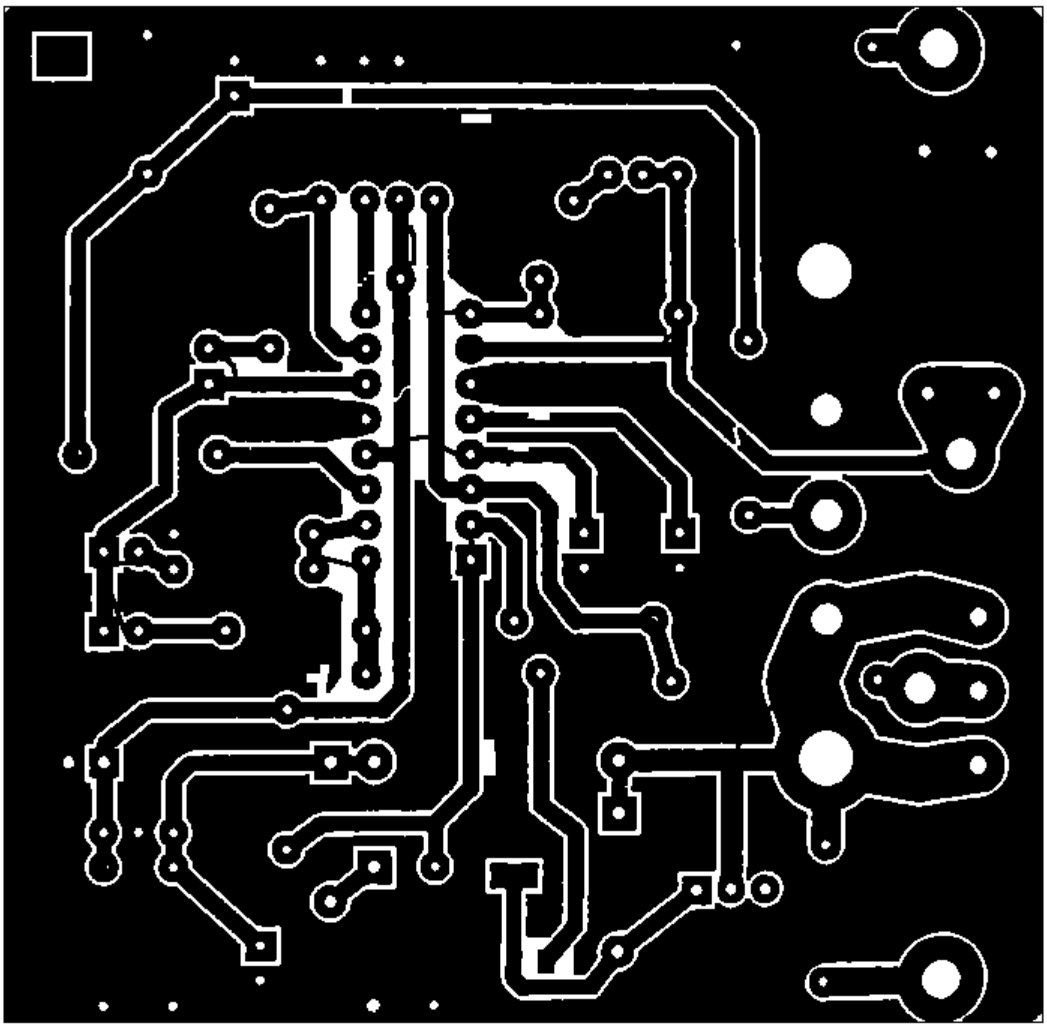
3.10 lentelė. Defektų detektavimo skaitinės reikšmės, panaudojus morfologinę opraciją *„erode“*

Defekto pavadinimas	Kiekis
Takelių netoleruotinų klaidų skaičius	3
Takelių toleruotinų klaidų skaičius	6
Trumpiklio netoleruotinų klaidų skaičius	5
Trumpiklio toleruotinų klaidų skaičius	34
Sienos netoleruotinų klaidų skaičius	2
Sienos toleruotinų klaidų skaičius	8
Lydviečių netoleruotinų klaidų skaičius	78
Lydviečių toleruotinu klaidų skaičius	0
Fono klaidų skaičius	37

3.11 lentelė. Defektų detektavimo skaitinės reikšmės skirtumas lygininat su etaloninėmis reikšmėmis panaudojus morfologinę opraciją *„erode“*

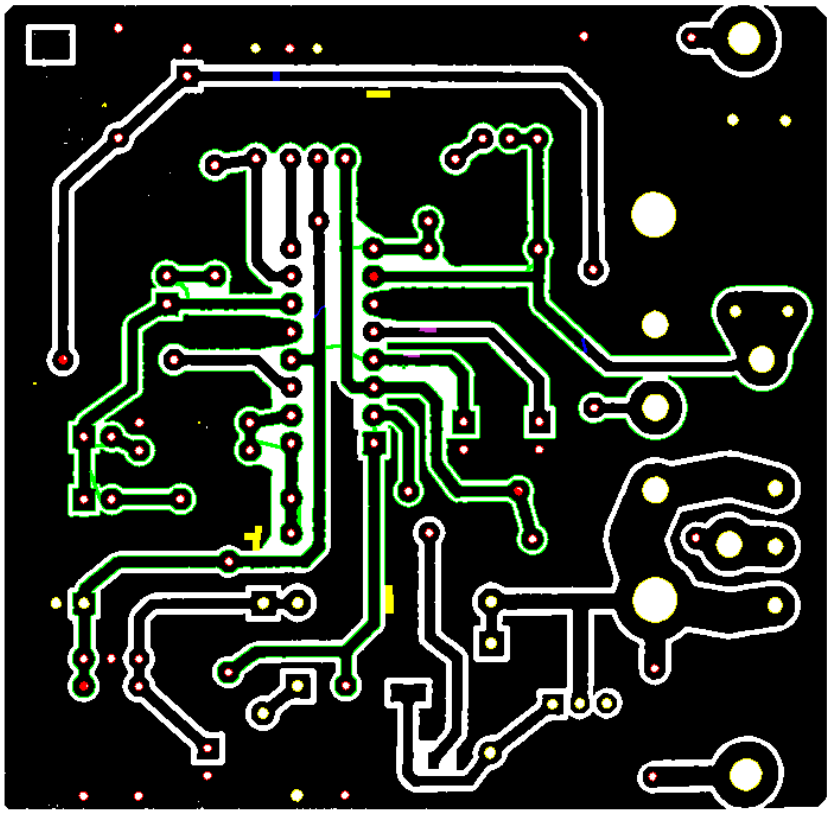
Defekto pavadinimas	Kiekis
Takelių netoleruotinų klaidų skaičius	-8
Takelių toleruotinų klaidų skaičius	-2313
Trumpiklio netoleruotinų klaidų skaičius	-3
Trumpiklio toleruotinų klaidų skaičius	-28
Sienos netoleruotinų klaidų skaičius	0
Sienos toleruotinų klaidų skaičius	-15
Lydviečių netoleruotinų klaidų skaičius	+73
Lydviečių toleruotinu klaidų skaičius	-1
Fono klaidų skaičius	+34

Po morfologinės operacijos „*erode“* panaudojimo, gautų skaitinių reikšmių skirtumas žymimas su minuso ženklu, jeigu naudojant šią funkciją gauti rezultatai, lyginant su etalonėmis skaitinėmis reikšmėmis, yra mažesni t. y., neatitikimų buvo rasta mažiau nei nurodo etaloninės skaitinės reikšmės. Jeigu skaitinių reikšmių skirtumas žymimas su pliuso ženklu, vadinasi neatitikimų padaugėjo. Sutapus defektų skaitinėms reikšmėms, žymima 0.

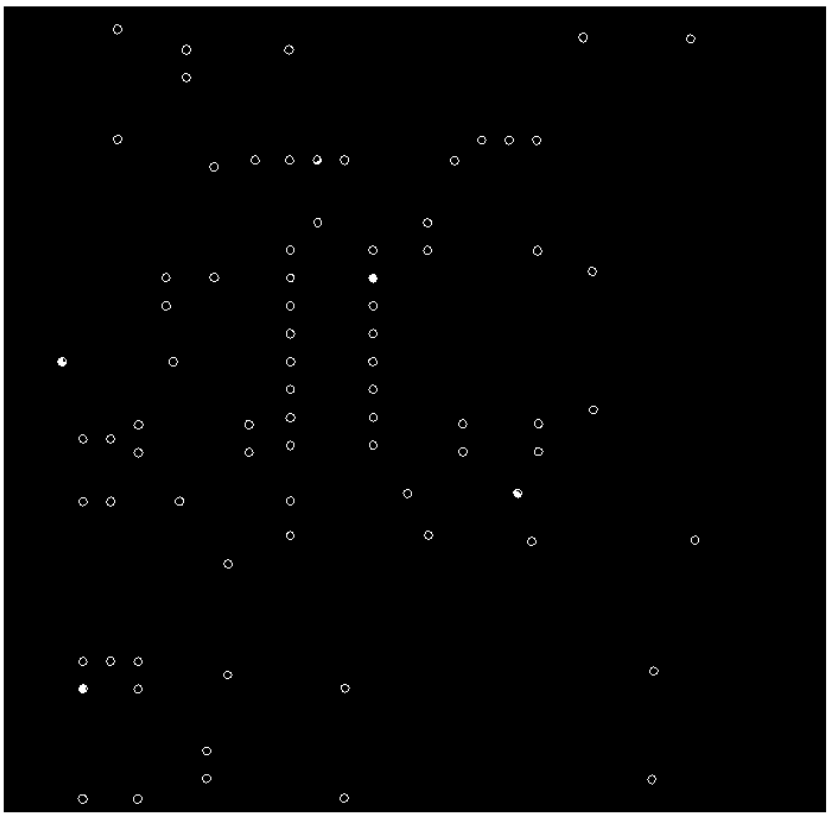


3.19 pav. Tikrinamos PCB atvaizdas po morfologinės operacijos „*erode“* panaudojimo

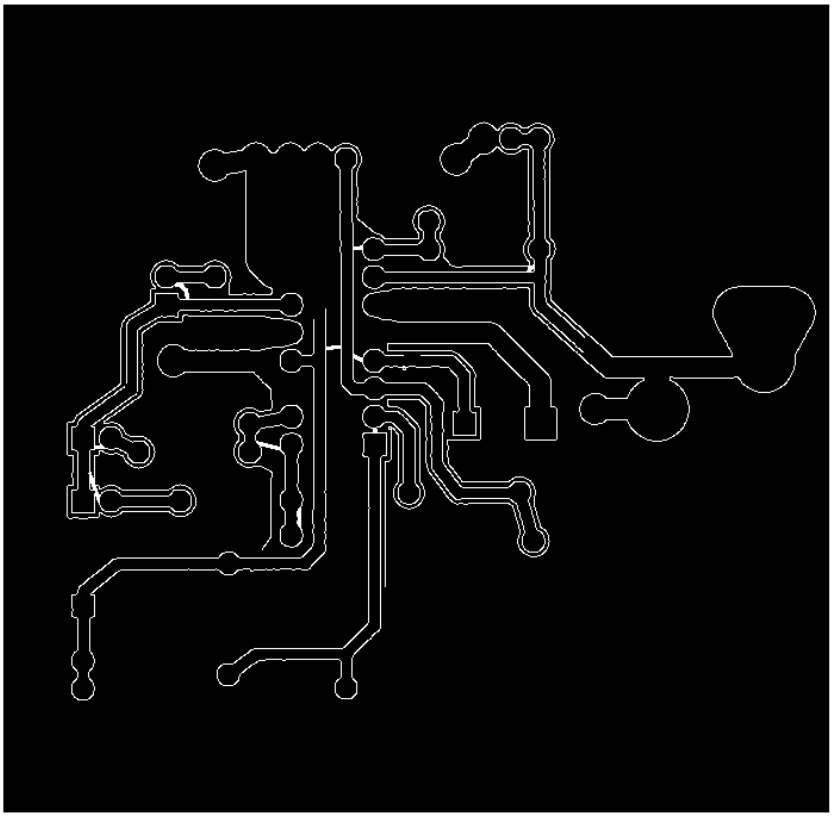
Po atvaizdo valymo, naudojant „*erode“* operaciją, žymiai sumažėjo takelių ir trumpiklių defektų, kadangi „*erode“* operacija „priaugino“ papildomų pikselių, taip pašalinant mažus takelių ir trumpiklių defektus. Dėl to, kad buvo „priauginami“, sumažėjo lydviečių plotas, ko pasekoje, visos lydvietės buvo priskirtos prie defektų.



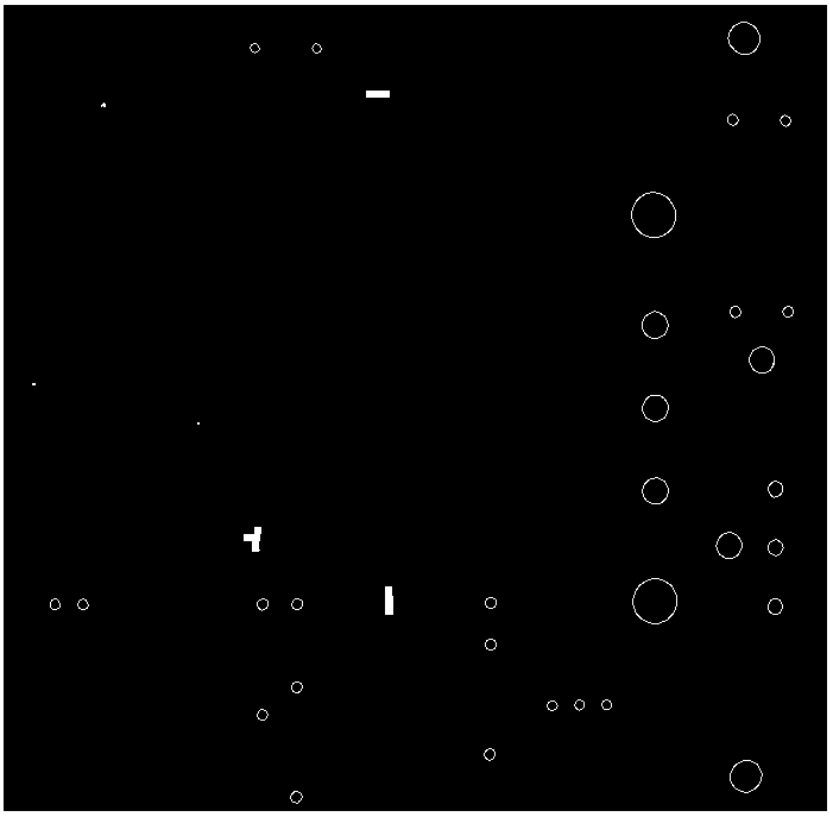
3.20 pav. Ištestuotas PCB atvaizdas po morfologinės operacijos „*erode“* panaudojimo



3.21 pav. Lydviečių defektai po po morfologinės operacijos „*erode“* panaudojimo



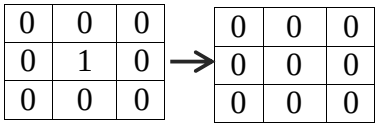
3.22 pav. Trumpiklių defektai po po morfologinės operacijos „*erode“* panaudojimo



3.23 pav. Fono klaidos po morfologinės operacijos „*erode“* panaudojimo

3.5 Morfologinės operacijos „clean“ naudojimas šalinant triukšmus

Binariniams vaizdmas apdoroti skirta morfologinė operacija - valymas (angl. *clean*), suvienodina, 3x3 matricoje, viduriniojo pikselio reikšmę su jį supančiai pikseliais (3.24 pav.)



3.24 pav. 3x3 matrica

3.12 lentelė. Defektų detektavimo skaitinės reikšmės, panaudojus morfologinę opraciją *„clean“*

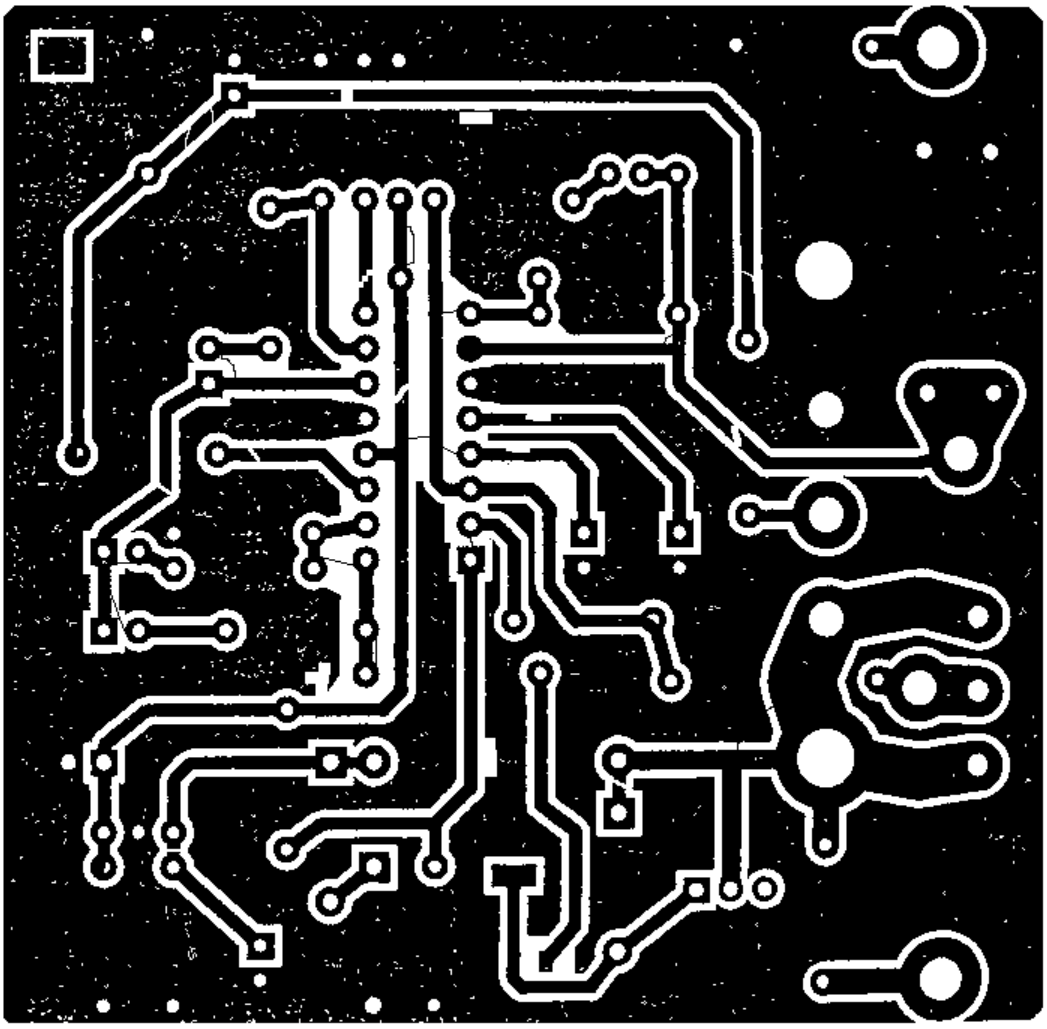
Defekto pavadinimas	Kiekis
Takelių netoleruotinų klaidų skaičius	11
Takelių toleruotinų klaidų skaičius	2344
Trumpiklio netoleruotinų klaidų skaičius	8
Trumpiklio toleruotinų klaidų skaičius	5
Sienos netoleruotinų klaidų skaičius	2
Sienos toleruotinų klaidų skaičius	23
Lydviečių netoleruotinų klaidų skaičius	5
Lydviečių toleruotinu klaidų skaičius	0
Fono klaidų skaičius	182

3.13 lentelė. Defektų detektavimo skaitinės reikšmės skirtumas lygininat su etaloninėmis reikšmėmis panaudojus morfologinę opraciją *„clean“*

Defekto pavadinimas	Kiekis
Takelių netoleruotinų klaidų skaičius	0
Takelių toleruotinų klaidų skaičius	+25
Trumpiklio netoleruotinų klaidų skaičius	0
Trumpiklio toleruotinų klaidų skaičius	-1
Sienos netoleruotinų klaidų skaičius	0
Sienos toleruotinų klaidų skaičius	0
Lydviečių netoleruotinų klaidų skaičius	0
Lydviečių toleruotinu klaidų skaičius	-1
Fono klaidų skaičius	+179

Po morfologinės operacijos „*clean“* panaudojimo, gautų skaitinių reikšmių skirtumas žymimas su minuso ženklu, jeigu naudojant šią funkciją gauti rezultatai, lyginant su etalonėmis skaitinėmis reikšmėmis, yra mažesni t. y., neatitikimų buvo rasta mažiau nei nurodo etaloninės skaitinės

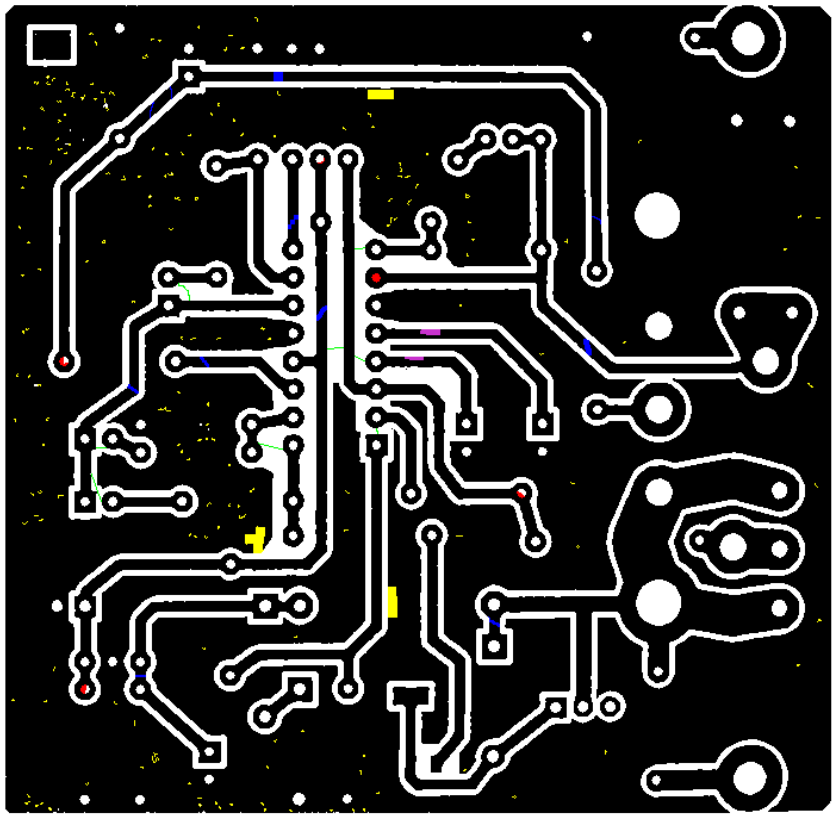
reikšmės. Jeigu skaitinių reikšmių skirtumas žymimas su pliuso ženklu, vadinasi neatitikimų padaugėjo. Sutapus defektų skaitinėms reikšmėms, žymima 0.



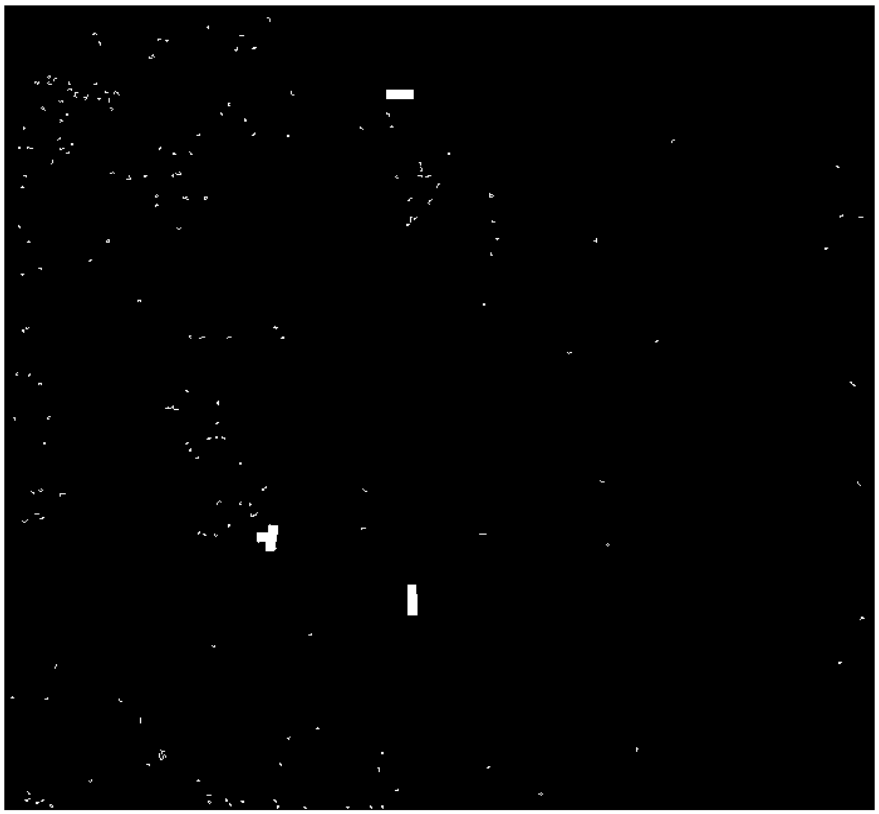
3.25 pav. Tikrinamos PCB atvaizdas po morfologinės operacijos „*clean“* panaudojimo

Panaudojus „*clean“* operaciją, išvalyta didelė dalis triukšmų (3.25 pav.). Po atlikto valymo matome, kad nebuvo pašalinti takelių smulkūs defektai kurių plotis ~1 pikselį.

Lyginant su etalonėmis skaitinėmis reikšmėmis, padidėjo fono klaidų skaičius. Papildomai po valymo, lyginant su etaloninėmis skaitinėmis reikšmėmis, sumažėjo trumpiklių ir lydviečių toleruotinos klaidos.



3.26 pav. Ištestuotas PCB atvaizdas po morfologinės operacijos „*clean*“ panaudojimo



3.27 pav. Fono klaidos po morfologinės operacijos „*clean*“ panaudojimo

3.6 Gautų rezultatų apibendrinimas

Išbandžius medianinį ir standartinio nuokrypio filtrus, bei morfologines operacijas „*Bwareaopen*“ su 4 ir 8 pikselių barjerais, „*erode*“ ir „*clean*“, gauti rezultatai yra skirtingi. Apibendrinant yra palyginami trijų defektų rūšių: *takelių įtrūkimas, trumpikliai, lydvietės,* detektavimo tikslumo, naudojant skirtingus triukšmo šalinimo būdus ir lyginant su etaloniniu defektų detektavimu.

3.14 lentelė. Takelio įtrūkimo defektų radimas, nurodytais filtrais/operacijomis apdorojus tikrinamą atvaizdą ir lyginant su etalonine defektų skaitine reikšme

	Takelių įtrūkimai	Skirtumas	Skirtumas %
Etaloninė reikšmė	11	±	
Medianos filtras	6	-5	45,5
Standartinio nuokrypio filtras	8	-3	27,3
Morf. operacija „*Bwareaopen*“ 4	11	0	0
Morf. operacija „*Bwareaopen*“ 8	11	0	0
Morf. operacija „*erode*“	3	-8	72,7
Morf. operacija „*clean*“	11	0	0

3.15 lentelė. Trumpiklių defektų radimas, nurodytais filtrais/operacijomis apdorojus tikrinamą atvaizdą ir lyginant su etalonine defektų skaitine reikšme

	Trumpikliai	Skirtumas	Skirtumas %
Etaloninė reikšmė	8	±	
Medianos filtras	0	-8	100
Standartinio nuokrypio filtras	2	-6	75
Morf. operacija „*Bwareaopen*“ 4	8	0	0
Morf. operacija „*Bwareaopen*“ 8	8	0	0
Morf. operacija „*erode*“	5	-3	37,5
Morf. operacija „*clean*“	8	0	0

3.16 lentelė. Lydviečių defektų radimas, nurodytais filtrais/operacijomis apdorojus tikrinamą atvaizdą ir lyginant su etalonine defektų skaitine reikšme

	Lydvietės	Skirtumas	Skirtumas %
Etaloninė reikšmė	5	±	
Medianos filtras	5	0	0
Standartinio nuokrypio filtras	78	+73	1560
Morf. operacija „*Bwareaopen*“ 4	5	0	0
Morf. operacija „*Bwareaopen*“ 8	5	0	0
Morf. operacija „*erode*“	78	+73	1560
Morf. operacija „*clean*“	5	0	0

Išvados ir rezultatai

1. Atlikus literatūros analizę, nustatyti pagrindiniai defektai kurie egzistuoja PCB gamyboje. Nustatyta defektų įtaka PCB plokštėms. Apžvelgtos PCB defektų inspektavimo technologijos, kurios naudojamos šiuolaikinėje pramonėje.
2. Aprašytas PCB gamybos procesas.
3. Pasiūlytas ir pilnai aprašytas, individualiai sukurtas, PCB defektų inspektavimo metodas, kuriuo remiantis inspektuojami takelių įtrūkimų, takelių iškandimų, trumpiklių ir lydviečių defektai.
4. Ištirti *Matlab* programiniame pakete siūlomų binarinio atvaizdo apdorojimo filtrų įtaka klaidų atpažinimo rezultatui, kai yra žinomi tikslūs defektai ir jų kiekis.
5. Apibendrinant tyrimą galiu teigti, jog užduoties realizavimui būtų užtekę morfologinių operacijų *Bwareaopen, erode, clean*. Naudojant šias morfologines operacijas, po atlikto triukšmų valymo, nei vienas defektas nebuvo įtakotas, nepasikeitė jų pikselinė struktūra, tačiau fono triukšmas pašalintas nepilnai. Nors triukšmas, esantis fone t. y. ant dialektrinio paviršiaus, kuris izoliuoja laidininkus, neįtakoja PCB kokybės, tačiau remiantis tuo, kad buvo siekta optimaliai pašalinti visus egzistavusius triukšmingus pikselius, individualiai sukurti filtrai, kurie išvalė beveik visus triukšmus.

Literatūros šaltiniai

[1] Dalius Navakauskas, Artūras Serackis „Skaitmeninis signalų apdorojimas taikant MATLAB“ Vilnius 2013.

[2] interneto prieiga: Local standard deviation of image, [žiūrėta 2016 04 20], http://se.mathworks.com/help/images/ref/stdfilt.html

[3] interneto prieiga: Remove small objects from binary image, [žiūrėta 2016 04 23] http://se.mathworks.com/help/images/ref/bwareaopen.html

[4] interneto prieiga: Printed Circuit Board manufacturing, [žiūrėta 2016 04 22] http://www.madehow.com/Volume-2/Printed-Circuit-Board.html

[5] interneto prieiga: PCB Inspection Techniques & Technologies, [žiūrėta 2016 05 02] http://www.radio-electronics.com/info/t_and_m/ate/electronics-pcb-inspection.php

[6] interneto prieiga: Basic Printed Circuit Board Manufucture, [žiūrėta 2016 05 02] http://www.smartgroup.org/downloads/Basic%20PCB%20Manufacture.pdf

[7] Donatas Dervinis „Vaizdų apdorojimas“ Kaunas 2012.

[8] interneto prieiga: Median filter, [žiūrėta 2016 03 29] http://homepages.inf.ed.ac.uk/rbf/HIPR2/median.htm

[9] interneto prieiga: Smoothing operations, [žiūrėta 2016 04 12] http://www.mif.vu.lt/atpazinimas/dip/FIP/fip-Smoothin.html

[10] Girūta Kazakevičiūtė – Januškevičienė „Rastrinių vaizdų geometrinės transformacijos“ Vilnius 2012.

[11] interneto prieiga: Pixel, [žiūrėta 2016 03 15], http://whatis.techtarget.com/definition/pixel

[12] T. Young, J Gerbrands „Fundamentals of Image Processing“ 2007.

[13] M. Kumar „DIGITAL IMAGE PROCESSING“ 2008

[14] M. H. Tibana, R. A. Lotufo „Novel Automatic PCB Inspection Technique Based on Connectivity“. In: X Brazilian Symph.on Computer Graphics and Image Processing – SIBGRAP’07, Campos de Jordao, Vol.1. 2007 p. 187-194.

[15] S. B. Gokturk, L. Akarun „Automated inspection of PCB‘s using a novel Approach“ 2005.

[16] T. Du-Ming, Y. Ron-Hwa „An eigenvalue-based similarity measure and its aplication in defect detection“, Image and Vision Computing, Vol. 23, No.12, 2006 p. 1094 – 1101

[17] F. R. Leta „Discussing Accuracy in an Automatic Measurement System using Computer Vision Techniques“ COBEM 2005.

[18] interneto prieiga: A. McAndrew „An Introduction to Digital Image Processing with Matlab“ [žiūrėta 2016 05 02]
https://eclass.teicrete.gr/modules/document/file.php/TM152/Lab/00_Intro/Matlab-Image_Processing_Tutorial.pdf

[19] interneto prieiga: AOI Light Source [žiūrėta 2016 04 25]
http://cyberoptics.com/pdf/aoi/sq3000/AddressingHighPrecisionAutomatedOpticalInspection.pdf

[20] A. P. S. Chauhan, S. C. Bhardwaj „Detection of Bare PCB Defects by Image Substraction Method using Machine Vision“, Proceedings of the World Congress on Engineering Vol. 2, 2011

[21] Greenberg „Method for printed circuit board inspection“. U.S.A pattent 6, 990, 227.

[22] interneto prieiga: Automated X-ray Inspection“ [žiūrėta 2016 05 11],
http://www.keysight.com/upload/cmc_upload/All/Why_Where_What_How_When.pdf?&cc=LT&lc=eng

Priedas Nr.1

Vaizdai

```
clear all;
close all;
Rotation=0; % Posukio kampas laipsniais
a = imread('blogaRGB.png');
gray= rgb2gray(a); %konvertuoja i nespalvota
BW = im2bw(gray); %konvertuoja i binarini

z = imread('geraRGB.png');
z2 = im2bw(z);
%z11=Valymas(z2,4);
[n,m]=size(BW);% suvienodinam paveikslu mamtmenis jei riekia
z1 = imresize(z2, [n m]);
%% Tikrinamo paveikslo posukio kampo nustatymas lyginant su ETALONU
zz1=z1(1:80,1:80);
vv1=BW(1:80,1:80);
zz=Valymas(zz1,4);
vv=Valymas(vv1,4);
 [Tkx Tky Tdx Tdy]=Kaimpai(zz);
 [T1kx T1ky T1dx T1dy]=Kaimpai(vv);
figure
imshow(zz)
 hold on
plot(Tkx, Tky, 'r*');
plot(Tdx,Tdy,'r*');
figure
imshow(vv)
 hold on
 plot(T1kx, T1ky, 'b*');
```

```
plot(T1dx,T1dy,'b*');


movingPoints = [T1kx T1ky;T1dx T1dy];% taskai kurie zymi zymes (kvadr) kampus
fixedPoints = [Tkx Tky;Tdx Tdy];

%% Randamas paveikslo posukio kampas, jei jis skiriasi nuo etalono
t = cp2tform(movingPoints,fixedPoints,'nonreflective similarity');
u = [0 1];
v = [0 0];
[x, y] = tformfwd(t, u, v);
dx = x(2) - x(1);
dy = y(2) - y(1);
angle = (180/pi) * atan2(dy, dx);
scale = 1 / sqrt(dx^2 + dy^2);
if angle~=0
Rotation=angle;
end
v1=imrotate(BW,Rotation);
%% Valome etalonini paveiksla
[n,m]=size(v1);% suvienodinam paveikslu mamtmenis jei riekia
z1 = imresize(z2, [n m]);
z11=Valymas(z1,4);% Valau etaloini paveiksla
%% Randam  Skirtingus ETALONINES schemos objektus, kanalus, lydvietes, pagrinda
Pav = bwlabel(z11, 8);
blobMeasurements = regionprops(Pav, 'Area', 'Perimeter', 'EulerNumber');
numberOfBlobs = size(blobMeasurements, 1);
allPerimeters = [blobMeasurements.Perimeter];
allAreas = [blobMeasurements.Area];
for i=1:length(allAreas)
   % Skaiciuojame apvalumo metrika
  metric(i) = 4*pi*allAreas(i)/allPerimeters(i)^2;
end
```

```
  % compute the roundness metric
%  metric = 4*pi*area/perimeter^2;
allEuler = [blobMeasurements.EulerNumber]; % Num regions - num holes in region.
circularities = allPerimeters .^ 2 ./ (4*pi*allAreas);
% Ieskoma sriciu kurios butu apskritimai
circularBlobIndexes = find(circularities < 2);
NotcircularBlobIndexes = find(metric <.85);
Kanalai = ismember(Pav, NotcircularBlobIndexes);
Trumpikliai=~Kanalai;

Lydvietes1 = ismember(Pav, circularBlobIndexes);

blobMeasurements = regionprops(Lydvietes1, 'EquivDiameter');
allDiameters = [blobMeasurements.EquivDiameter];
%----
k=1;
for i=1:length(allDiameters)
  if allDiameters(i)>6.75 && allDiameters(i)<8.25
     IndexD(k)=circularBlobIndexes(i);
     Diametrai(k)= allDiameters(i);
     k=k+1;
   end
end
%  figure
 Lydvietes = ismember(Pav,IndexD);
%
%  imshow(Lydvietes)
%  figure
%  imshow(~Trumpikliai)
%% randam tikrinamo paveikslo lydvieciu diametrus:
Pavv = bwlabel(~z11, 8);
blobMeasurements1 = regionprops(Pavv, 'Area', 'Perimeter', 'EulerNumber');
allAreas1 = [blobMeasurements1.Area];
% Ieskoma didziauios plotu srities
```

```
BlobIndexes1 = find(allAreas1<max(allAreas1));
% Isskiriami kanalai
KanalaiTikri = ismember(Pavv,BlobIndexes1);
 figure
 imshow(KanalaiTikri)
 title('Kanalai tikri');
%% Valome neEtalonini paveiksla
v2=Valymas2(BW,4,KanalaiTikri);
v22 = bwareaopen(BW,4); % Matlabo fitras naikinantis
figure
imshow(v22)
title('bwareaopen filruotas paveikslas 4 pix ');
v221 = bwareaopen(BW,8); % Matlabo fitras naikinantis
figure
imshow(v221)
title('bwareaopen filruotas paveikslas 8 pix');
v222 = medfilt2(BW);
figure
imshow(v222)
title('medfilt2 filruotas paveikslas ');
v2222 = stdfilt(BW);
figure
imshow(v2222)
title('stdfilt filruotas paveikslas ');
v22222 = bwmorph(BW,'clean');
figure
imshow(v22222)
title('bwmorph "clean" filruotas paveikslas ');
v222222= bwmorph(BW,'erode');
figure
imshow(v222222)
title('bwmorph "erode" filruotas paveikslas ');
v2222222= bwmorph(BW,'dilate');
figure
```

```
imshow(v2222222)
title('bwmorph "dilate" filruotas paveikslas ');
%% Kliadu ieskojimas
[n,m]=size(v2);
z1 = imresize(z11, [n m]);
KlaidLydvietes(1:n,1:m)=0;%klaidu paveiksliukas
KlaidKanalai(1:n,1:m)=0;%klaidu paveiksliukas
KlaidTrumpikliai(1:n,1:m)=0;%klaidu paveiksliukas
KlaidPagrindas(1:n,1:m)=0;%klaidu paveiksliukas
SienosKlaidos(1:n,1:m)=0;
%TKlaid(1:n,1:m)=0;% toleruotinu klaidu paveiksliukas
eil=n;% eiluciu skaiciu
stulp=m;% stulpeliu skaicius
for i=2:eil-1
   for j=2:stulp-1
      if z1(i,j)~=v2(i,j) && Lydvietes(i,j)==1
      KlaidLydvietes(i,j)=1;
      end
        if z1(i,j)~=v2(i,j) && KanalaiTikri(i,j)==1
      KlaidKanalai(i,j)=1;
        end
        if z1(i,j)~=v2(i,j) && Kanalai(i,j)==1
      KlaidTrumpikliai(i,j)=1;
        end
        if z1(i,j)~=v2(i,j) && Kanalai(i,j)~=1 && KanalaiTikri(i,j)~=1 && Lydvietes(i,j)~=1
      KlaidPagrindas(i,j)=1;
        end
   end
end

%% Klaidu klaisfikavimas
Pix=4; % didziausios toleruotinos klaidos dydis pixeliais
[KlaiduPavP,KlaiduskP,KdydisP,KpavP]=Jungumas(KlaidPagrindas);
[KlaiduPavL,KlaiduskL,KdydisL,KpavL]=Jungumas(KlaidLydvietes);
```

```
[KlaiduPavK,KlaiduskK,KdydisK,KpavK]=Jungumas(KlaidKanalai);
[KlaiduPavT,KlaiduskT,KdydisT,KpavT]=Jungumas(KlaidTrumpikliai);

[NetolKlaidP,NetolKlaidpavP,NPaP,NTKlaidsP,TolKlaidosP,TolKlaidpavP,PTPaP,TKlaidsP]=Klai
duKlase(KlaiduPavP,KpavP,KdydisP,Pix);
[NetolKlaidL,NetolKlaidpavL,NPaL,NTKlaidsL,TolKlaidosL,TolKlaidpavL,TPaL,TKlaidsL]=Klai
duKlase(KlaiduPavL,KpavL,KdydisL,Pix);
[NetolKlaidK,NetolKlaidpavK,NPaK,NTKlaidsK,TolKlaidosK,TolKlaidpavK,TPaK,TKlaidsK]=Kl
aiduKlase(KlaiduPavK,KpavK,KdydisK,Pix);
[NetolKlaidT,NetolKlaidpavT,NPaT,NTKlaidsT,TolKlaidosT,TolKlaidpavT,TPaT,TKlaidsT]=Klai
duKlase(KlaiduPavT,KpavT,KdydisT,Pix);
%% Tikslinam trupiklio sienos ir kanalo kliadas bei ju kritiskuma
Pavv = bwlabel(~v2, 8);
blobMeasurements1 = regionprops(Pavv, 'Area', 'Perimeter', 'EulerNumber');
allAreas1 = [blobMeasurements1.Area];
BlobIndexes1 = find(allAreas1<max(allAreas1));
KanalaiTNeetalono = ismember(Pavv,BlobIndexes1);
%-----------------------------------------------------
Pav = bwlabel(v2, 8);
blobMeasurements = regionprops(Pav, 'Area', 'Perimeter', 'EulerNumber');
allPerimeters= [blobMeasurements.Perimeter];
allAreas = [blobMeasurements.Area];
for i=1:length(allAreas)
   % Skaiciuojame apvalumo metrika
  metric(i) = 4*pi*allAreas(i)/allPerimeters(i)^2;
end
NotcircularBlobIndexes = find(metric <.85);
KanalaiNET = ismember(Pav, NotcircularBlobIndexes);
TrumpikliaiNeetalono=~KanalaiNET;
%---------------- Sienos Kanalo ir trumpiklio kritiniu klaidu ieskojimas----------------------------------
--
%[KanNeT,numKNeT]=bwlabel(Trumpikliai);
[KaneT,numKeT]=bwlabel(KanalaiTikri);
[TrumNeT,numTNeT]=bwlabel(~TrumpikliaiNeetalono);
```

```
[TrumeT,numeTeT]=bwlabel(~Trumpikliai);
[KanNeT,numKNeT]=bwlabel(~TrumNeT);
figure
imshow(KanNeT)
figure
imshow(TrumNeT)
[eil,stulp]=size(KanNeT);
SienosKlaidos(1:eil,1:stulp)=0;
a=0;%pagalbiniai kintamieji
b=0;%
for i=1:eil
  for j=1:stulp

      if NetolKlaidK(i,j)==1 && (z1(i-1,j)~=z1(i+1,j)|| z1(i,j-1)~=z1(i,j+1))
      Siena1(i,j)= 1;
      end
  end
end
SienosKLaidSk=0;
SienosKLaidSkTol=0;
%[KlaiduPavS,SienosKLaidSkTol,KdydisS,KpavS]=Jungumas(Siena1);
% Sienos klaidu klasifikavimas
[P,nn] = bwlabel(Siena1, 4);
Matai = regionprops(P, 'Area');
SritciuDydziai = [Matai.Area];
%Koordinates=Matai.Extrema;
d=1;
[KlaiduPavS,SienosKLaidSkTol,KdydisS,KpavS]=Jungumas(P);
for z=1:length(KdydisS)
    if KdydisS(z)>8 % renkiesi pagal situacija
 [x,y]=find(KlaiduPavS==KpavS(z));
 Skaicius=NetolKlaidpavK(x(1),y(1));
    [ro,co]=find(NetolKlaidpavK==Skaicius);
     Skaic=NetolKlaidpavK(ro(1),co(1));
```

```
    for d=1:length(ro)
     SienosKlaidos(ro(d),co(d))=1;
     NetolKlaidK(ro(d),co(d))=0;
     %TolKlaidosK(ro(d),co(d))=0;
     Siena1(ro(d),co(d))=0;
     end
     SienosKLaidSk=SienosKLaidSk+1;
     NTKlaidsK=NTKlaidsK-1;
     %TKlaidsK=TKlaidsK-1;
    [~,MM] =find(NPaK==Skaic);
     NPaK(MM)=[];
    end

end

SienosKLaidSkTol=SienosKLaidSkTol-SienosKLaidSk;%toleruotinu sienos klaidu skaicius
[n,m]=size(KlaiduPavK);
% Kanalo Klaidu klasifikavimas
for j=1:length(NPaK)
        [r, c] = find(KlaiduPavK==NPaK(j));
           a=0;
           b=0;
         for i=1:length(r)

              if r(i)<n && c(i)<m && KanNeT(r(i)+1,c(i)+1)~=0 && a==0
              a=KanNeT(r(i)+1,c(i)+1);
              end
              if r(i)>1 && c(i)<m && KanNeT(r(i)-1,c(i)+1)~=0 && a==0
              a=KanNeT(r(i)-1,c(i)+1);
              end
              if r(i)>1 &&  c(i)>1 &&  KanNeT(r(i)-1,c(i)-1)~=0 && a==0
              a=KanNeT(r(i)-1,c(i)-1);
              end
              if c(i)>1 && r(i)<n &&  KanNeT(r(i)+1,c(i)-1)~=0 && a==0
```

```
a=KanNeT(r(i)+1,c(i)-1);
end
 if c(i)>1 &&  KanNeT(r(i),c(i)-1)~=0 && a==0
a=KanNeT(r(i),c(i)-1);
 end
 if c(i)<m &&  KanNeT(r(i),c(i)+1)~=0 && a==0
a=KanNeT(r(i),c(i)+1);
 end
  if r(i)>1 &&  KanNeT(r(i)-1,c(i))~=0 && a==0
a=KanNeT(r(i)-1,c(i));
 end
 if r(i)<n &&  KanNeT(r(i)+1,c(i))~=0 && a==0
a=KanNeT(r(i)+1,c(i));
end
%%%%%%%%
if r(i)<n && c(i)<m && KanNeT(r(i)+1,c(i)+1)~=0 && a~=0 && a~=KanNeT(r(i)+1,c(i)+1) && b==0
b=KanNeT(r(i)+1,c(i)+1);
end
if r(i)>1 && c(i)<m && KanNeT(r(i)-1,c(i)+1)~=0 && a~=0 && a~=KanNeT(r(i)-1,c(i)+1) && b==0
b=KanNeT(r(i)-1,c(i)+1);
end
if r(i)>1 && c(i)>1 && KanNeT(r(i)-1,c(i)-1)~=0 && a~=0 && a~=KanNeT(r(i)-1,c(i)-1) && b==0
b=KanNeT(r(i)-1,c(i)-1);
end
if r(i)<n && c(i)>1 && KanNeT(r(i)+1,c(i)-1)~=0 && a~=0 && a~=KanNeT(r(i)+1,c(i)-1) && b==0
b=KanNeT(r(i)+1,c(i)-1);
end
 if c(i)<m && KanNeT(r(i),c(i)+1)~=0 && a~=0 && a~=KanNeT(r(i),c(i)+1) && b==0
b=KanNeT(r(i),c(i)+1);
end
```

```
            if c(i)>1 && KanNeT(r(i),c(i)-1)~=0 && a~=0 && a~=KanNeT(r(i),c(i)-1) && b==0
            b=KanNeT(r(i),c(i)-1);
            end
            if r(i)>1 && KanNeT(r(i)-1,c(i))~=0 && a~=0 && a~=KanNeT(r(i)-1,c(i)) && b==0
            b=KanNeT(r(i)-1,c(i));
            end
            if r(i)<n && KanNeT(r(i)+1,c(i))~=0 && a~=0 && a~=KanNeT(r(i)+1,c(i)) && b==0
            b=KanNeT(r(i)+1,c(i));
            end

        end

         if a==0 || b==0 %|| a==b
         NTKlaidsK=NTKlaidsK-1; % Perskaiciuojami klaidu skaiciai
         TKlaidsK=TKlaidsK+1;
          for k=1:length(r)
         NetolKlaidK(r(k),c(k))=0; % Pasalinamos nekiritines klaidos
         TolKlaidosK(r(k),c(k))=1;
          end
         end

end
% Trumpiklio klaidu klasifikavimas
for j=1:length(NPaT)

         [r, c] = find(KlaiduPavT==NPaT(j));
          a=0;
          b=0;
  for i=1:length(r)


            if r(i)<n && c(i)<m && KaneT(r(i)+1,c(i)+1)~=0 && a==0
            a=KaneT(r(i)+1,c(i)+1);
            end
```

```
if r(i)>1 && c(i)<m && KaneT(r(i)-1,c(i)+1)~=0 && a==0
a=KaneT(r(i)-1,c(i)+1);
end
if r(i)>1 && c(i)>1 && KaneT(r(i)-1,c(i)-1)~=0 && a==0
a=KaneT(r(i)-1,c(i)-1);
end
if c(i)>1 && r(i)<n && KaneT(r(i)+1,c(i)-1)~=0 && a==0
a=KaneT(r(i)+1,c(i)-1);
end
 if c(i)>1 && KaneT(r(i),c(i)-1)~=0 && a==0
a=KaneT(r(i),c(i)-1);
 end
 if c(i)<m && KaneT(r(i),c(i)+1)~=0 && a==0
a=KaneT(r(i),c(i)+1);
 end
  if r(i)>1 && KaneT(r(i)+1,c(i)-1)~=0 && a==0
a=KaneT(r(i)-1,c(i));
 end
 if r(i)<n && KaneT(r(i)+1,c(i)-1)~=0 && a==0
a=KaneT(r(i)+1,c(i));
end
%%%%%%%%
if r(i)<n && c(i)<m && KaneT(r(i)+1,c(i)+1)~=0 && a~=0 && a~=KaneT(r(i)+1,c(i)+1)
&& b==0
b=KaneT(r(i)+1,c(i)+1);
end
if r(i)>1 && c(i)<m && KaneT(r(i)-1,c(i)+1)~=0 && a~=0 && a~=KaneT(r(i)-1,c(i)+1)
&& b==0
b=KaneT(r(i)-1,c(i)+1);
end
if r(i)>1 && c(i)>1 && KaneT(r(i)-1,c(i)-1)~=0 && a~=0 && a~=KaneT(r(i)-1,c(i)-1)
&& b==0
b=KaneT(r(i)-1,c(i)-1);
end
```

```
            if r(i)<n && c(i)>1 && KaneT(r(i)+1,c(i)-1)~=0 && a~=0 && a~=KaneT(r(i)+1,c(i)-1)
&& b==0
            b=KaneT(r(i)+1,c(i)-1);
            end
            if c(i)<m && KaneT(r(i),c(i)+1)~=0 && a~=0 && a~=KaneT(r(i),c(i)+1) && b==0
            b=KanNeT(r(i),c(i)+1);
            end
            if c(i)>1 && KaneT(r(i),c(i)-1)~=0 && a~=0 && a~=KaneT(r(i),c(i)-1) && b==0
            b=KanNeT(r(i),c(i)-1);
            end
            if r(i)>1 && KaneT(r(i)-1,c(i))~=0 && a~=0 && a~=KaneT(r(i)-1,c(i)) && b==0
            b=KanNeT(r(i)-1,c(i));
            end
            if r(i)<n && KaneT(r(i)+1,c(i))~=0 && a~=0 && a~=KaneT(r(i)+1,c(i)) && b==0
            b=KanNeT(r(i)+1,c(i));
            end

  end


        if a==0 || b==0 %|| a==b
        NTKlaidsT=NTKlaidsT-1; % Perskaiciuojami klaidu skaiciai
        TKlaidsT=TKlaidsT+1;
         for k=1:length(r)
        NetolKlaidT(r(k),c(k))=0; % Pasalinamos nekiritines klaidos
        TolKlaidosT(r(k),c(k))=1;

         end
        end

end

%% Paveikslu braizymas
 figure,imshow(z11,[]),title('Etaloninis schemos paveikslas ');
```

```
 figure,imshow(v2,[]),title('Tikrinamos schemos paveikslas ' );
 figure,imshow(NetolKlaidK,[]),title('Netoleruotinos kanalo klaidos: ');
 figure,imshow(TolKlaidosK,[]),title('Toleruotinos kanalo klaidos: ' );
 figure,imshow(NetolKlaidT,[]),title('Netoleruotinos Trumpiklio klaidos: ');
 figure,imshow(TolKlaidosT,[]),title('Toleruotinos Trumpiklio klaidos: ' );
 figure,imshow(SienosKlaidos,[]),title('Netoleruotinos sienos klaidos: ');
 figure,imshow(Siena1,[]),title('Toleruotinos sienos klaidos: ' );
 figure,imshow(NetolKlaidL,[]),title('Netoleruotinos Lydvieciu klaidos: ');
 figure,imshow(TolKlaidosL,[]),title('Toleruotinos Lydvieciu klaidos: ' );
 figure,imshow(NetolKlaidP,[]),title('Netoleruotinos Fono klaidos: ');
 figure,imshow(TolKlaidosP,[]),title('Toleruotinos Fonoo klaidos: ' );

 % Klaidu ir originalaus paveikslo spalvojimas

 grayImage = uint8(255 * z1); % dvinario paveikslo vertimas i pilka
[rows, columns, numberOfColorChannels] = size(grayImage);

binaryImage = v2;
% pilko paveikslo vertimas i rgb
if numberOfColorChannels < 3
                rgbImage = cat(3, grayImage, grayImage, grayImage);
else

                rgbImage = grayImage;
end
% Spalvu kanalu isfiltravimas
redChannel = rgbImage(:, :, 1);
greenChannel = rgbImage(:, :, 2);
blueChannel = rgbImage(:, :, 3);
% Spalvu kuriomis noresime spalvoti tam tikras vietas parinkimas
desiredColor = [0, 0, 255]; % Melyna KANALO spalva
desiredColor1 = [0, 255, 0]; % Zalia TRUMPIKLIO ospalva
desiredColor2 = [255,0, 0]; %Raudona LYDVIETES spalva
desiredColor3 = [255, 255, 0]; % geltona Pagrindo spalva
```

```
desiredColor4 = [200,50,205]; %Rozine SIENOS spalva
% Vietu spalvojimas
 for i=1:eil
    for j=1:stulp
     if NetolKlaidK(i,j)~=0
redChannel(i,j) = desiredColor(1);
greenChannel(i,j) = desiredColor(2);
blueChannel(i,j) = desiredColor(3);
     end
     if NetolKlaidT(i,j)~=0
redChannel(i,j) = desiredColor1(1);
greenChannel(i,j) = desiredColor1(2);
blueChannel(i,j) = desiredColor1(3);
     end
     if NetolKlaidL(i,j)~=0
redChannel(i,j) = desiredColor2(1);
greenChannel(i,j) = desiredColor2(2);
blueChannel(i,j) = desiredColor2(3);
     end
       if NetolKlaidP(i,j)~=0
redChannel(i,j) = desiredColor3(1);
greenChannel(i,j) = desiredColor3(2);
blueChannel(i,j) = desiredColor3(3);
       end
       if SienosKlaidos(i,j)~=0
redChannel(i,j) = desiredColor4(1);
greenChannel(i,j) = desiredColor4(2);
blueChannel(i,j) = desiredColor4(3);
      end
    end
 end
% skirtingu kanalu grazinimas i viena pilka paveiksla
rgbImage = cat(3, redChannel, greenChannel, blueChannel);
```

```
figure,imshow(rgbImage,[]),title('Pilkas schemos paveikslas su klaidomis, melynos trumpiklio, geltonos kanalo, raudonos sienos klaidos ')
%% rezultatu isvedimas
 A(1,1)=cellstr('Kanalo netoleruotinu klaidu skaicius');
 A(2,1)=cellstr('Kanalo toleruotinu klaidu skaicius');
 A(3,1)=cellstr('Trumpiklio netoleruotinu klaidu skaicius');
 A(4,1)=cellstr('Trumpiklio toleruotinu klaidu skaiciu');
 A(5,1)=cellstr('Sienos netoleruotinu klaidu skaicius');
 A(6,1)=cellstr('Sienos toleruotinu klaidu skaicius');
 A(7,1)=cellstr('Lydvieciu netoleruotinu klaidu skaicius');
 A(8,1)=cellstr('Lydvieciu toleruotinu klaidu skaicius');
 A(9,1)=cellstr('Pagrindo netoleruotinu klaidu skaicius');
 A(10,1)=cellstr('Pagrindo toleruotinu klaidu skaicius');
 A(1,2)=num2cell(NTKlaidsK);
 A(2,2)=num2cell(TKlaidsK);
 A(3,2)=num2cell(NTKlaidsT);
 A(4,2)=num2cell(TKlaidsT);
 A(5,2)=num2cell(SienosKLaidSk);
 A(6,2)=num2cell(SienosKLaidSkTol);
 A(7,2)=num2cell(NTKlaidsL);
 A(8,2)=num2cell(TKlaidsL);
 A(9,2)=num2cell(NTKlaidsP);
 A(10,2)=num2cell(TKlaidsP);
 filename ='C:\Users\Vartotojas\Desktop\Ats.xls';
 xlswrite(filename,A);
```

Vaizdų valymas

```
function Y=Valymas(X,KL)
[eil,stulp]=size(X);
Y=X;
kl=0;
for i=2:eil-1% ejnam per eilute
   for j=2:stulp-1% einam per stulpelius
       if X(i,j)==1% jeigu baltas

        for k=-1:2:1
          if X(i+k,j)==0 % Sklaiciuojam klaidas
          kl=kl+1;
          end
          if X(i,j+k)==0
          kl=kl+1;
          end
          if X(i+k,j+k)==0
          kl=kl+1;
          end
          if X(i+k,j-k)==0
          kl=kl+1;
          end
        end
          if kl>=KL
          Y(i,j)=0; % jei klaidu barjeras virsijamas pikselis baltas
          else
          Y(i,j)=1; % jei ne lieka baltas
          end
       end
          kl=0;
   end
end
```

Takelių valymas

```
function Y=Valymas2(X,KL,Z)
[eil,stulp]=size(X);
Y=X;
kl=0;
for i=2:eil-1
   for j=2:stulp-1 % tas pats kaip Valymas
       if X(i,j)==1
         for k=-1:2:1
          if X(i+k,j)==0
          kl=kl+1;
          end
          if X(i,j+k)==0
          kl=kl+1;
          end
          if X(i+k,j+k)==0
          kl=kl+1;
          end
          if X(i+k,j-k)==0
          kl=kl+1;
          end
         end
        if Z(i,j)==0
           if kl>=KL
            Y(i,j)=0;
           else
             Y(i,j)=1;
           end
        end
         if Z(i,j)==1
           if kl>=7 % padidintas klaidu barjeras kanalui
            Y(i,j)=0;
           else
```

```
            Y(i,j)=1;
          end
        end


      end
         kl=0;
  end
end
```

Priskyrimas

```
function Y=Pryskirimas(L,num2,num)
%[eil,stulp]=size(L);
[r, c] = find(L==num2);
%rc = [r c];
%[n,m]=size(rc);
 for i=1:length(r)
   %for j=1:m
  L(r(i),c(i))=num;
 end

Y=L;
```

Klaidų klasifikavimas

```
function [X,Xp,Xx,xsk,Y,Yp,Yy,ysk]=KlaiduKlase(KlaiduPav,Kpav,Kdydis,pix)
[eil,stulp]=size(KlaiduPav);
Y(eil,stulp)=0; % Toleruotinos klaidos
X(eil,stulp)=0; % Netoleruotinos klaidos
Yp(eil,stulp)=0; % Toleruotinuklaidu pavadinimai
Xp(eil,stulp)=0; % Netoleruotinu klaidu pavadinimai
Xx=0;
Yy=0;
```

```
xsk=0;%klaidu Netoleruotinu skaicius
ysk=0;%klaidu Toleruotinu skaicius

t=1;
tt=1;
for k=1:length(Kdydis)
 [r, c] = find(KlaiduPav==Kpav(k));
    if Kdydis(k)>=pix

     Xx(t)=Kpav(k);
     t=t+1;
  for i=1:length(r)

     X(r(i),c(i))=1;
     Xp(r(i),c(i))=Kpav(k);

  end
  xsk=xsk+1;
    end
    if Kdydis(k)<pix

      Yy(t)=Kpav(k);
      tt=tt+1;
   for i=1:length(r)

     Y(r(i),c(i))=1;
     Yp(r(i),c(i))=Kpav(k);

   end

 ysk=ysk+1;
    end
end
```

Kampų nustatymas

```
function [Tkx Tky Tdx Tdy]=Kaimpai(Pav11)% ieskom kampo koord
[eil,stulp]=size(Pav11);
Tkx=0;
Tky=0;
Tdx=0;
Tdy=0;

for i=3:eil-2
   for j=3:stulp-2
if Pav11(i,j)==0 && Pav11(i+1,j)==1 && Pav11(i,j+1)==1 && Pav11(i+1,j+1)==1 && Pav11(i-1,j-1)==0 && Pav11(i-1,j)== 0 && Pav11(i+1,j-1)==0 && Pav11(i-1,j+1)==0 && Pav11(i-1,j)==0 && Pav11(i+2,j)==1 && Pav11(i,j+2)==1
     Tkx=j;
     Tky=i;
end
if Pav11(i,j)==0 && Pav11(i,j-1)==1 && Pav11(i+1,j-1)==1 && Pav11(i+1,j)==1 && Pav11(i+1,j+1)~=1 && Pav11(i,j+1)~= 1 && Pav11(i-1,j+1)==0 && Pav11(i-1,j)==0 && Pav11(i-1,j-1)~=1 && Pav11(i+2,j)==1 && Pav11(i,j-2)==1
     Tdx=j;
     Tdy=i;
end

   end
end
```

Jungumas

```
function [Y,N,Kdydis,KPav]=Jungumas(X)
[eil,stulp]=size(X);
Y(eil,stulp)=0;
N=1;
[L,num] = bwlabel(X);
```

```
for k=1:num
[r, c] = find(L==k);
rc = [r c];
[n,m]=size(rc);
  for i=2:n-1
    for j=2:m-1
        Y(i,j)=num;
         for k=-1:2:1
           if L(i+k,j)~=0 && L(i+k,j)~=L(i,j)
           num2=L(i+k,j);
           L=Pryskirimas(L,num2,num);
           end
           if L(i,j+k)~=0 &&  L(i,j+k)~=L(i,j)
           num2=L(i+k,j);
           L=Pryskirimas(L,num2,num);
           end
           if L(i+k,j+k)~=0 && L(i+k,j+k)~=L(i,j)
           num2=L(i+k,j);
           L=Pryskirimas(L,num2,num);
           end
           if L(i+k,j-k)~=0 && L(i+k,j-k)~=L(i,j)
           num2=L(i+k,j);
           L=Pryskirimas(L,num2,num);
           end
         end


     end
   end
end
Y=L;
k=0;
for i=1:num
```

```
    [row, col] = find(Y==i);
    if length(row)~=0;
    k=k+1;
    Kdydis(k)=max([length(row) length(col)]);
    KPav(k)=i;
    end
end
N=k;
```