Kauno technologijos universitetas

Informatikos fakultetas

Modulis „T120M023 Programų testavimo metodai“

Testavimo planas

IFM 9/2 gr. Algirdas Kartavičius

Studentas

Doc. Šarūnas Packevičius

Dėstytojas

Kaunas, 2020

Turinys

1. Įvadas

Šis testavimo planas yra skirtas baigiamojo magistrinio darbo sistemos testavimui. Baigiamojo magistrinio darbo tema yra susijusi su kompiuterinės regos ir mašininio mokymosi pritaikymo tyrimu pramoninės gamybos gaminių kokybės patikrai. Planuojama kurti sistemos dalį, kuri bus integruota į bendrą baldų patikros sistemą. Darbe planuojama įsigilinti į baldų detalių paviršiaus defektų aptikimą ir klasifikavimą.

Šiuo metu labai daug gamybos procesų yra automatizuojami, nemažoje dalyje stambesnių gamyklų yra diegiami robotizacijos sprendimai. Nors šios automatizuotos sistemos turi labai daug privalumų lyginant su žmonių darbu, tačiau jos turi ir keletą trūkumų. Automatizuotos sistemos dažnai yra labai sudėtingos ir susideda iš daugybės komponentų, todėl yra gana didelė tikimybė, kad tam tikras komponentas suges. Toks gedimas gali turėti labai rimtų padarinių – visi pagaminti produktai bus su defektais ir bus iššvaistyta dar daugiau žaliavų. Šios rimtos problemos galima išvengti panaudojant kompiuterinės regos sprendimus, kurie užtikrina gaminių kokybę bei stabdo gamybos procesą, jei tam tikras netinkamų produktų kiekis pagaminamas iš eilės. Vizualinės patikros privalumai pramoninėje gamyboje yra labai ženklūs, nes produktų kiekiai yra labai dideli ir net nedidelis kokybės pagerinimas leidžia sutaupyti daug pinigų. Kompiuterine rega pagrįsti produktų patikros metodai yra plačiai ištirti ir naudojami pagerinti produktų kokybę bei sumažinti išlaidas. Beveik visada pramoninių gaminių kokybės patikra vyksta realiu laiku, todėl reikia ne tik analizuoti gautą iš nuotraukų vaizdą, bet ir užtikrinti, kad gaunamas vaizdas būtų kokybiškas ir tinkamas defektų aptikimui.

Vykdant projektus, kurie yra susiję su kompiuterine rega, neapsieinama be aparatūrinės įrangos. Beveik visų projektų įgyvendinimui reikia kamerų arba tam tikrų jutiklių. Programinės įrangos, kurioje yra naudojama aparatūrinė įranga kelia nemažai iššūkių, dažniausiai programinė įranga gali būti ištestuota tik specifinėse tam skirtose patalpose, tačiau panaudojus šiuolaikinius automatinius programinės įrangos testus nemažai funkcionalumo galima ištestuoti ir be įrangos, imituojant gaunamą iš jutiklių arba naudojant jau išsaugotus skaitmeninius vaizdus. Didžioji dalis gamybos kokybės patikros sistemų turi sugebėti patikrinti gaminius realiu laiku, kadangi tai svarbu norint išlaikyti aukštą efektyvumą. Baldų detalių patikra išsiskiria tuo, kad gaminių matmenys yra gana dideli, tačiau reikia pastebėti pakankamai smulkius defektus. Baldų detalių patikrai dažnai naudojamos linijos nuskaitymo kameros, todėl gaunami labai didelės skiriamosios gebos vaizdai. Vykdant analizę realiu laiku labai svarbus skaitmeninių vaizdų apdorojimo algoritmų greitaveikos testavimas.

2. Testavimo tikslai

Sistemos testavimo tikslas yra atskleisti kuo daugiau programos klaidų bei padaryti sistemos veikimą stabilesniu. Taip pat sistemos testavimas padeda sumažinti riziką, kad atliekant programos atnaujinimus ar pakeitimus atsiras klaidų, kurių anksčiau nebuvo. Testavimas taip pat leis sukurti aukštesnės kokybės sistemos versiją bei užtikrinti, kad atlikus atnaujinimus sistema ir toliau stabiliai veiks.

3. Testavimo apimtis

Testavimo metu planuojama atlikti vienetų, integravimo, priėmimo, greitaveikos bei aukšto lygio testavimą. Kuriant vieneto testus bus užtikrinta, kad kiekvienas kuriamos sistemos dalies metodas bus padengiamas paduodant tinkamus bei netinkamus duomenis. Vykdant integravimo testavimą bus užtikrinama, kad sistemos klasės tinkamai sąveikauja viena su kita. Taip bus užtikrinama duomenų kontrolė. Priėmimo testavimas bus atliekamas vykdant standartinius sistemos naudotojo veiksmus. Vykdant aukšto lygio testavimą bus atliekamas sistemos testavimas su realiomis baldų detalėmis bei skaitmeninėmis vaizdo kameromis. Aukšto testavimo metu ypatingai bus skiriama dėmesio sistemos našumui bei greitaveikai. Vykdant greitaveikos testavimą bus tikrinamas algoritmų efektyvumas su didelės skaitmeninės skiriamosios gebos vaizdais. Greitaveikos testavimo metu bus testuojamas ir aparatūrinės kompiuterio įrangos našumas, kadangi taikant mašininio mokymosi algoritmus didelę įtaką greitaveiką turi grafinio procesoriaus skaičiavimų greitis.

4. Testavimo resursai ir apribojimai

Kadangi skaitmeninių vaizdų apdorojimas reikalauja gana daug resursų, todėl nuspręsta nustatyti minimalius testavimo aplinkos resursus. Taip pat baldų patikros sistema yra sukurta naudojant „C#" programavimo kalbą bei „.NET framework" karkasą, todėl mašininio mokymosi algoritmai turi veikti su „Windows" operacine sistema.

Minimalūs testavimo aplinkos resursai:

- Windows 10 operacinė sistema;
- 16 GB operatyviosios atminties;
- 250 GB talpos *SSD* diskas;
- 8 branduolių procesorius;
- Grafinis procesorius („GEFORCE RTX 2080" atitinka šiuos parametrus):
    - 11 GB „VRAM";
    - 14 „TFLOPs" skaičiavimo resursų;
    - 4000 „NVIDIA CURA" branduolių.

5. Testavimo strategija

5.1. Vienetų testavimas

Vienetų testai bus kuriami sistemos klasėms bei jų metodams. Vienetų testai vykdomi paduodant tam tikrus duomenis į metodas. Į metodus bus paduodami tinkami bei netinkami duomenys. Bus tikrinama ar metodų grąžinami rezultatai sutampa su vertėmis, kurių yra tikimasi. Vienetų testavimą planuojama atlikti programavimo metu, kadangi planuojama taikyti „TDD“ principą.

5.2. Integravimo testavimas

Sukūrus vienetų testus, pašalinus kritines klaidas bei įsitikinus, kad visi metodai grąžina tinkamus rezultatus bus kuriami integravimo testai. Planuojama naudoti daugiasluoksnį testavimo principą („bottom-up“ ir „top-down“ testavimo principų kombinacija). Integravimo testavimo metu bus patikrinama sąveika tarp skirtingų sistemos komponentų ir bus patikrinama ar bendras komponentų sąveikos rezultatas yra toks kokio yra tikimasi.

5.3. Priėmimo testavimas

Priėmimo testavimo metu bus atliekami pagrindiniai sistemos naudotojo žingsniai. Testavimo metu bus peržiūrimas pagrindinis sistemos funkcionalumas bei patikrinama ar šis funkcionalumas atitinka sistemos užsakovo poreikius. Radus neatitikimų specifikacijoje arba sistemos veikime šie neatitikimai bus taisomi.

5.4. Aukšto lygio testavimas

Aukšto lygio testavimas bus atliekamas prie realaus testavimo stendo, kurį sudaro kompiuteris su našiu grafiniu procesoriumi, ritininis konvejeris, kuris skirtas baldų detalių tolygiam transportavimui, vaizdo kameros, programuojamas loginis valdiklis bei kita aparatūrinė įranga.

5.5. Testavimo įrankiai

Planuojama naudoti integruotą programavimo aplinką „Visual studio 2019“. Testų vykdymui nuspręsta panaudoti integruotos programavimo aplinkos įskiepį „Resharper“. Patys testai bus kuriami panaudojus atvirojo kodo testavimo karkasą „xUnit“.

6. Testavimas

6.1. Vienetų testavimas

Žemiau pateikiami vienetų testų atvejai, kurias planuojama testuoti pagrindinius defektų aptikimo metodus.

1 lentelė Klasės „MachineLearningDefectDetection“ metodo „AnalyzePartOfImage“ testavimo atvejai

Testas	**Rezultatas**
Vykdomas „AnalyzePartOfImage“ metodas kai parametras „img“ yra inicializuotas bet neturi jokios informacijos (nėra vaizdo informacijos).	Metodas sustoja ties vaizdo informacijos tikrinimu, grąžinama tuščias aptiktų defektų sąrašas.
Vykdomas „AnalyzePartOfImage“ metodas kai parametras „img“ yra „null“.	Metodas sustoja ties vaizdo informacijos tikrinimu, grąžinama tuščias aptiktų defektų sąrašas.
Vykdomas „AnalyzePartOfImage“ metodas kai parametras „img“ yra tinkamai inicializuotas bei užpildytas vaizdo informacija.	Metodas sėkmingai įvykdomas, grąžinamas sąrašas su aptiktais defektais.

2 lentelė Klasės „MachineLearningDefectDetection“ metodo „MatToBytes“ testavimo atvejai

Testas	**Rezultatas**
Vykdomas „MatToBytes“ metodas kai parametras „img“ yra inicializuotas bet neturi jokios informacijos (nėra vaizdo informacijos).	Metodas sustoja ties vaizdo informacijos tikrinimu, grąžinama „null“ reikšmė.
Vykdomas „MatToBytes“ metodas kai parametras „img“ yra „null“.	Metodas sustoja ties vaizdo informacijos tikrinimu, grąžinama „null“ reikšmė.
Vykdomas „MatToBytes“ metodas kai parametras „img“ yra tinkamai inicializuotas bei užpildytas vaizdo informacija.	Metodas sėkmingai įvykdomas grąžinama vaizdo informacija paversta į baitų masyvą.

3 lentelė Klasės „MachineLearningDefectDetection“ metodo „DrawRectangles“ testavimo atvejai

Testas	**Rezultatas**
Vykdomas „DrawRectangles“ metodas kai parametras „img“ yra inicializuotas bet neturi jokios informacijos (nėra vaizdo informacijos).	Metodas sustoja ties vaizdo informacijos tikrinimu, grąžinama „null“ reikšmė.

Vykdomas „DrawRectangles“ metodas kai parametras „img“ yra „null“.	Metodas sustoja ties vaizdo informacijos tikrinimu, grąžinama „null“ reikšmė.
Vykdomas „DrawRectangles“ metodas kai parametras „img“ yra tinkamai inicializuotas bei užpildytas vaizdo informacija, vaizdas yra trijų kanalų (spalvotas).	Metodas sėkmingai įvykdomas, grąžinama vaizdas su pažymėtais defektais.
Vykdomas „DrawRectangles“ metodas kai parametras „img“ yra tinkamai inicializuotas bei užpildytas vaizdo informacija, vaizdas yra vieno kanalo (nespalvotas).	Vaizdas paverčiamas į trijų kanalų, metodas sėkmingai įvykdomas, grąžinama vaizdas su pažymėtais defektais.

4 lentelė Klasės „MachineLearningDefectDetection“ metodo „AnalyzeImage“ testavimo atvejai

Testas	**Rezultatas**
Vykdomas „AnalyzeImage“ metodas kai parametras „img“ yra inicializuotas bet neturi jokios informacijos (nėra vaizdo informacijos).	Metodas sustoja ties vaizdo informacijos tikrinimu, grąžinama tuščias aptiktų defektų sąrašas.
Vykdomas „AnalyzeImage“ metodas kai parametras „img“ yra „null“.	Metodas sustoja ties vaizdo informacijos tikrinimu, grąžinama tuščias aptiktų defektų sąrašas.
Vykdomas „AnalyzeImage“ metodas kai parametras „img“ yra tinkamai inicializuotas bei užpildytas vaizdo informacija.	Metodas sėkmingai įvykdomas, grąžinamas sąrašas su aptiktais defektais.

5 lentelė Klasės „MachineLearningDefectDetection“ metodo „AnalyzeImageFromFile“ testavimo atvejai

Testas	**Rezultatas**
Vykdomas „AnalyzeImageFromFile“ metodas kai parametras „filename“ yra neegzistuojantis failas	Metodas sustoja ties failo egzistavimo tikrinimu, grąžinama tuščias aptiktų defektų sąrašas.
Vykdomas „AnalyzeImageFromFile“ metodas kai parametras „filename“ yra egzistuojantis failas	Metodas sėkmingai įvykdomas, grąžinamas sąrašas su aptiktais defektais.

6.2. Integravimo testavimas

Atlikus vienetų testavimą ir įsitikinus, kad pagrindinės sistemos dalys neturi kritinių klaidų bus pradedamas integravimo testavimas. Pasirinktas daugiasluoksnio testavimo metodas, kadangi kiti metodai pasirodė mažiau efektyvūs. Naudojant „bottom-up“ arba „top-down“ testavimo metodus tam

tikrų sistemos dalių integravimo testavimas pradedamas tik labai vėlai. Pasirinkto testavimo metodo principas yra pradėti testavimą viduriniame sluoksnyje ir tuo pačiu testuoti žemesnį bei aukštesnį sluoksnius. Šis metodas reikalauja daugiau resursų bei žinių tačiau leidžia pasiekti didesnį patikimumą.

6.3. Priėmimo testavimas

Priėmimo testavimo metu sistemos naudotojai tikrins ar sistema atitinka reikalavimų specifikaciją bei tinkamai atlieka visas funkcijas. Priėmimo testavimo metu bus įvertinama ir grafinės naudotojo sąsajos funkcionalumas bei aiškumas. Visi priėmimo testavimo dalyvaujantys naudotojai užpildys apklausą, kurioje galės įvertinti sistemos funkcionalumą bei išsakyti savo pastabas

6.4. Aukšto lygio testavimas

Aukšto lygio testavimo metu sistema bus testuoja su realiais baldų detalių pavyzdžiais. Bus bandoma su įvairių matmenų baldų detalėmis, kadangi pagal detalių dydį priklauso apdorojamos skaitmeninių vaizdų informacijos kiekis. Baldų detalių defektų aptikimo algoritmas bus testuojamas su įvairiai grafiniais procesoriais, kurie yra prieinami pasinaudojus „Google cloud“ debesų paslauga. Testavimas su skirtingais grafiniais procesoriais yra svarbu, kadangi gali nuspręsti kiek resursų reikia norint taikyti mašininio mokymosi algoritmus defektų aptikimui realiu laiku. Įsitikinus, kad mašininio mokymosi algoritmai yra pakankamai efektyvūs bus testuojama pasinaudojant testavimo stendu, kuris imituoja baldų gamyklos sąlygas. Baldų detalės bus transportuojamos konvejeriu, iš skaitmeninių vaizdo kamerų gaunamas vaizdas bus apdorojamas testavimo stendo kompiuteryje.

7. Išvados

1. Sudarant testavimo planą buvo suformuluoti testavimo tikslai, apimtis bei apribojimai, todėl testavimo procesas taps efektyvesniu ir greitesniu.
2. Buvo pasirinktas „xUnit“ testavimo karkasas, kadangi su šiuo karkasu yra testuojama ir visa likusi sistemos dalis bei jis yra plačiai naudojamas programų sukurtų su „C#” programavimo kalba testavimui.
3. Buvo nustatyti minimalūs testavimo aplinkos resursai, todėl tai leis geriau įvertinti sistemos testavimo aplinkos kainą.
4. Ypatingai daug dėmesio nuspręsta skirti aukšto lygio testavimui, kadangi sistema naudoja nemažai aparatūrinės įrangos.
5. Buvo nuspręsta išbandyti mašininio mokymosi algoritmų efektyvumą pasinaudojant „Google cloud“ platforma, kadangi ši paslauga suteikia prieigą prie gana galingų grafinių procesorių, kurie reikalingi bandymams, už sąlyginai mažą kainą. Mašininio algoritmo bandymas su įvairiais grafiniais procesoriais leis geriau nuspręsti kokį grafinį vaizdo procesorių reikia įsigyti norint, kad įdiegta sistema identifikuotų paviršiaus defektus realiu laiku.