Automatizirana optička analiza procesa razvoja tiskanih pločica

Vlahović, Stjepan

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Stjepan Vlahović

Zagreb, 2021.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentori:

Dr. sc. Tomislav Stipančić, dipl. ing.

Student:

Stjepan Vlahović

Zagreb, 2021.

Izjavljujem da sam ovaj rad izradio samostalno koristeći znanja stečena tijekom studija i navedenu literaturu.

Zahvaljujem se mentoru Dr. sc. Tomislav Stipančić, dipl. ing. na puno strpljenja i pomoći koju koju mi je pružio tokom izrade diplomskog rada.

Također bi se želio zahvaliti svojim roditeljima, djevojci, bratu, prijateljima i kolegama iz Smart Sense-a na razumijevanju i pomoći koju su mi pružili kroz studiranje.

Stjepan Vlahović

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za diplomske radove studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo materijala te mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum:	Prilog:
Klasa: 602-04/21-6/1	
Ur. broj: 15-1703-21	

DIPLOMSKI ZADATAK

Student: **STJEPAN VLAHOVIĆ** Mat. br.: 0035199416

Naslov rada na hrvatskom jeziku: **Automatizirana optička analiza procesa razvoja tiskanih pločica**

Naslov rada na engleskom jeziku: **Automated optical analysis of the printed circuit board development process**

Opis zadatka:

Elektroničke komponente često se povezuju u kompaktne cjeline pomoću tiskanih pločica (engl. printed circuit boards) na kojima se međusobno spajaju postupkom lemljenja. Sam proces lemljenja mora biti izveden precizno jer i najsitnija greška može prouzročiti kvar komponente ili cjeline. Osim ručnog postupka spajanja razvijene su i tehnologije automatizirane montaže komponenti (engl. surface-mount technology - SMT).
U radu je potrebno razraditi rješenje za sustav automatizirane optičke inspekcije procesa razvoja tiskanih pločica (engl. automated optical inspection – AOI), uključujući sljedeće:

1. predvidjeti te identificirati greške i probleme koji se mogu pojaviti kod automatizirane izrade tiskanih pločica,
2. osmisliti te opisati proizvodnu liniju za razvoj tiskanih pločica koja se temelji na AOI tehnologiji,
3. osmisliti sustav za automatizirano optičko otkrivanje grešaka,
4. osmišljeno rješenje je potrebno implementirati na realnom sustavu te eksperimentalno evaluirati.

U radu je potrebno navesti korištenu literaturu te eventualno dobivenu pomoć

Zadatak zadan:
6. svibnja 2021.

Zadatak zadao:
doc. dr. sc. Tomislav Stipančić

Rok predaje rada:
8. srpnja 2021.

Predviđeni datum obrane:
12. srpnja do 16. srpnja 2021.

Predsjednica Povjerenstva:
prof. dr. sc. Biserka Runje

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

POPIS TEHNIČKE DOKUMENTACIJE

202106250001 Adapter_01

202106250001 Adapter_02

POPIS OZNAKA

Oznaka	Jedinica	Opis
ϕ	/	Polarna koordinata u ϕ ravnini
ω	/	Parametar minimizacije
σ	/	Širina Gaussiana
$\boldsymbol{Z}$	/	Set vrijednosti intenziteta za *thresholding*
z	/	Koordinata u z ravnini
y	/	Koordinata u y ravnini
x	/	Koordinata u x ravnini
n	/	Broj epoha treniranja algoritma
f	m	Fokalna duljina leće
d	m	Udaljenost do točke u sceni
c	/	Originalna vrijednost piksela
b	/	Gornja granica u rasponu piksela
a	/	Donja granica u rasponu piksela
Y	/	Podatak iz baze za trening algoritma
T	/	Vrijednost intenziteta za *thresholding*
R	/	Regija slike
Q	/	Minimizirana funkcija
P	/	Mjera za homogenost
N	/	Površina kernela
M	/	Ukupni broj piksela u površini kernela
L	W/m^2	Zračenje scene
I	W/m^2	Zračenje po jedinici čvrstog kuta
G	/	Vrijednost gradijenta
F	/	Vrijednost novog piksela prilikom binarnog *thresholding*-a
E	W/m^2	Ozračenje točke
Θ	/	Polarna koordinata u Θ ravnini
q_i	/	Broj piksela u željenom histogramu
p_i	/	Broj piksela u originalnom histogramu
d'	m	Udaljenost ravnine slike od optičkog ishodišta kamere
c_k	/	Gornja granica u ekspandiranom rasponu piksela
c_1	/	Donja granica u ekpandiranom rasponu piksela
c'	/	Nova vrijednost piksela
$\hat{Y}$	/	Predviđena vrijednost algoritma
S_p	/	Set piksela
I_o	W/m^2	Zračenje površine iz generalnog izvora svjetlosti

SAŽETAK

U ovom radu se osmislila proizvodna linija s tehnologijom automatskog montiranja (SMT) te su se obradile greške koje se mogu javiti prilikom rada iste. S obzirom na najčešće greške koje se javljaju u praksi, u osmišljenu SMT proizvodnu liniju nadodao se sustav automatske optičke inspekcije (AOI). Međutim, dodatak novog sustava u već postojeću proizvodnu liniju nije jednostavan pothvat. Iz tog razloga se razvio i implementirao algoritam, na bazi konvolucijskih neuronskih mreža (CNN), koji prepoznaje specifičnu grešku.

Ključne riječi: tiskana elektronička pločica (PCB), montirana tiskana elektronička pločica (PCBA), SMT, AOI, CNN, Python, Raspberry Pi 3, Raspberry Pi kamera, CNC, UCCNC, Stepcraft-2/D.420

SUMMARY

In this paper, a production line with surface mounting technology (SMT) was designed and the errors that may occur during its operation were analyzed. Given the most common errors that occur in practice, an automatic optical inspection system (AOI) was added to the designed SMT production line. However, adding a new system to an existing production line is not an easy undertaking. For this reason, an algorithm based on convolutional neural networks (CNN), which recognizes a specific error, has been developed and implemented.

Key words: printed circuit board (PCB), printed circuit board assembly (PCBA), SMT, AOI, CNN, Python, Raspberry Pi 3, Raspberry Pi camera, CNC, UCCNC, Stepcraft-2/D.420

1. UVOD

Čovjekovom željom za konstantnim inovacijama došlo je do razvoja različitih tehnologija i uređaja. Svaki uređaj, kako bi mogao raditi ono za što je dizajniran, unutar sebe mora imati elektroničke komponente spojene preko tiskane pločice (PCB, engl. *Printed Circuit Board*).

Kako bi elektroničke komponente s PCB-om činile jednu cjelinu one se na istu leme, u procesu koji se naziva PCB montaža (PCBA, engl. *Printed Circuit Board Assembly*). Proces lemljenja je način spajanja metalnih dijelova s pomoću rastaljenog metalnog vezivnog sredstva. Navedeni proces se može raditi ručno (rad s malim serijama i većim komponentama) ili strojno.

U strojnom radu cilj je dobiti bržu i što konzistentniju izradu. U tu svrhu su razvijene tehnologije površinskog montiranja (SMT, engl. *Surface-mount technology*) koje automatiziraju cijeli proces.

Cilj ovog diplomskog rada je osmisliti SMT postrojenje, identificirati greške do kojih može doći prilikom rada te razviti i implementirati algoritam za automatsku optičku inspekciju (AOI, engl. *Automated optical inspection*) kojim se te greške mogu spriječiti ili ukloniti.

2. TEORIJSKA PODLOGA

Teorija bitna za razumijevanje rješenja diplomskog rada sadržana je u ovom poglavlju. Objašnjene su osnove SMT i AOI tehnologija te konvolucijske neuronske mreže (CNN, *engl. Convolutional Neural Network*) koje će se koristiti za izradu AOI algoritma.

2.1. SMT

SMT podrazumijeva tehnologiju montaže elektroničkih komponenti (SMD, engl. *Surface Mount Device*) na površinu PCB-a [1]. Tehnologija je razvijena s ciljem smanjenja troškova i vremena proizvodnje, kao i povećanja iskoristivosti površine PCB-a.

2.1.1. SMT proces

Proces započinje s dizajnom PCB-a i odabirom odgovarajućih komponenti koristeći software poput Altium designer-a [2] ili Eagle-a [3]. Po završetku dizajniranja, podaci za izradu PCB-a šalju se na proizvodnju, a komponente se zasebno kupuju kod dobavljača (npr. Mauser, TME, itd.). PCB-i dolaze na panelima (jedan panel sadrži više PCB-a), dok komponente mogu doći u više formata: koluti (engl. *reel*), tube (engl. *tube*), ladice (engl. *tray*), itd.

Ostatak procesa sastoji se od: SMT programiranja, printanja lemne paste (SPP, engl. *Solder Paste Printing*), inspekcije lemne paste (SPI, engl. *Solder Paste Inspection*), pozicioniranja komponenti, automatske optičke inspekcije prije *reflow* lemljenja [1] (engl. *Reflow soldering*), inspekcije prvog članka (FAI, engl. *First Article Inspection*), *Reflow* lemljenja, automatske optičke inspekcije poslije *reflow* lemljenja, selektivnog lemljenja (engl. *Selective Soldering*) i verifikacije procesa koristeći RTG inspekciju.

1. SMT programiranje

   Programirati se moraju: sustav za printanje lemne paste, svi sustavi za inspekciju te sustavi za optičku inspekciju. Svaki od sustava se može programirati *online* ili *offline*[2]. Kako bi se sustavi programirali potrebno je imati *gerber*[3], CAD podatke kao i *Pick and Place*

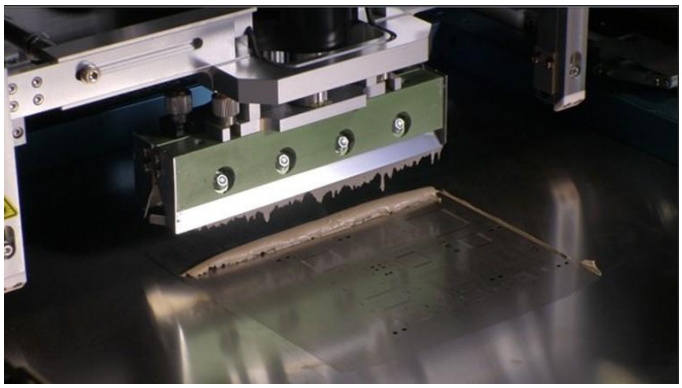
[1] *Reflow* lemljenje uključuje privremeno pričvršćivanje komponenti pastom za lemljenje, nakon čega se sklop podvrguje kontroliranoj toplini kako bi se postignuo trajni lemni sloj [4].
[2] *Offline* programiranje se radi na računalu nakon čega se program prenosi na sustav, dok se *online* programiranje radi direktno na sustavu.
[3] *Gerber* datoteka – ASCII vektorski format koji sadrži informacije svakog fizičkog sloja PCB-a [5].

datoteku[4] (PPF, eng, *Pick and Place file*). Programe je najlakše izraditi iz CAD podataka, no oni često nisu dostupni. Drugi način izrade programa je pomoću *gerber* podataka. Ukoliko se radi o programiranju sustava za postavljanje komponenti potrebno je imati i PPF.

Sustavi za lemljenje programiraju se *online*, za što je prethodno potrebno napraviti temperaturni profil koji će omogućiti optimalno rastapanje lemne paste i ponovno lemljenje.

2. Printanje i inspekcija lemne paste

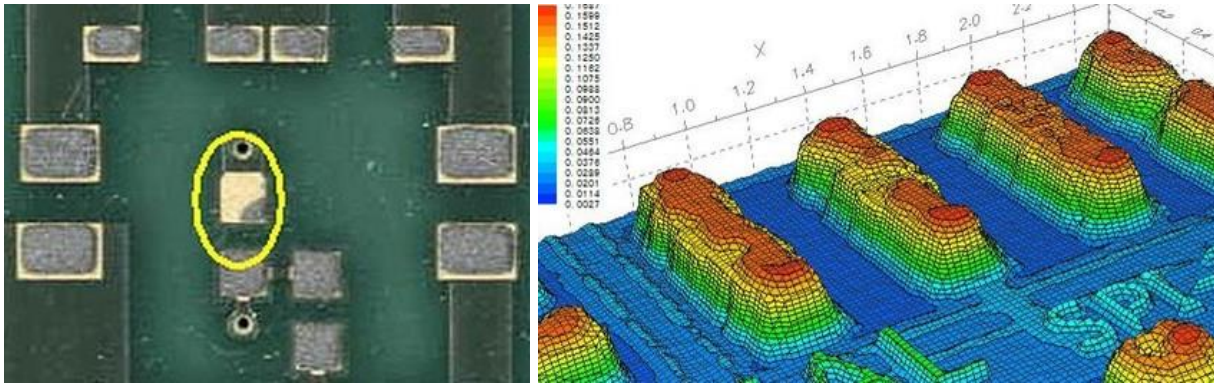
Prvi sustav koji se koristio za printanje lemne paste koristio je tehnologiju kojom se *squeegee*-jem[5] preko maske (engl. *stencil*) istišće lemna pasta na PCB panel, tzv. printanje preko maske (engl. *stencil printing*) (Slika 1). Maska se mora poklapati s pozicijama vodećih nožica (engl. *pads*), što regulira vizijski sustav za pozicioniranje maske i panela. Druga tehnologija je tehnologija printanja lemne paste mlaznim pritiskom (engl. *jet printing*). Tehnika izbacuje sitne kapljice lemne paste, kroz mehanizam za izbacivanje, na PCB panel prema *gerber* podacima koje primi prilikom programiranja. *Jet printing* tehnologija je pogodna za manje serije jer omogućava brzu izmjenu dva različita PCB projekta, dok je *stencile printing* tehnologija pogodnija za velike serije pošto zahtjeva masku za svaki zasebni PCB projekt. Budući da *stencile printing* tehnologija u velikim serijama može postići znatno veću brzinu printanja od *jet printing*-a, ona se i dalje se koristi u proizvodnji.



Slika 1. Printanje lemne paste [1]

Inspekcija lemne paste je do neke razine dostupna i unutar sustava za printanje lemne paste (integrirana inspekcija), no ukoliko proizvodnja zahtjeva veću sigurnost dodaje se SPI sustav. Za razliku od integrirane inspekcije, SPI sustav za detaljniju inspekciju koristi 3D tehnologiju koja osim površine, provjerava i volumen paste po vodećoj nožici (Slika 2).

[4] *Pick and place* datoteka – datoteka koja sadrži podatke s nazivom, pozicijama i orijentacijama komponenti.
[5] *Squeegee* – alat sa ravnom i glatkom oštricom koji se koristi za kontrolu toka na ravnim površinama.

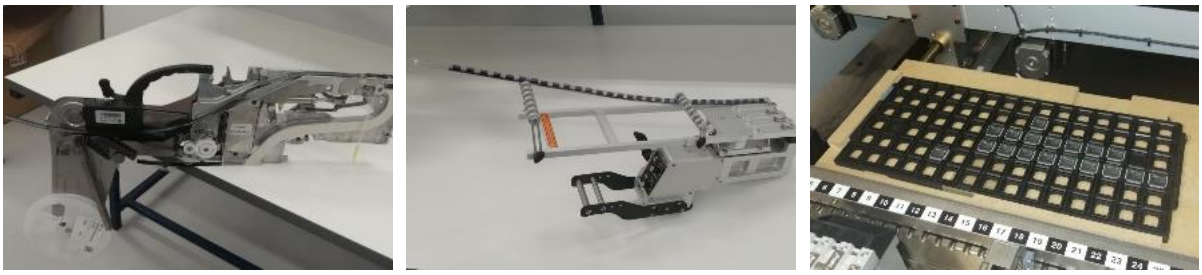


Slika 2. Inspekcija lemne paste – 2D (lijevo) i 3D (desno) [1]

3. Pozicioniranje komponenti

Ovaj proces podrazumijeva dohvaćanje svake komponente iz pakiranja, a ovisno o vrsti pakiranja razlikuju se načini dovođenja komponenti na poziciju izuzimanja:

- komponente u kolutima se na mjesto izuzimanja dovode tzv. „hranilicama“ (engl. *feeders*),
- komponente u tubama se dovode pomoću vibracijskih hranilica,
- dok se komponente u ladicama dovode poznavanjem rasporeda komponenti unutar ladice (Slika 3).



Slika 3. Hranilica (lijevo), vibracijska hranilica (sredina) i ladica (desno)

Sa mjesta izuzimanja komponente se mogu izuzeti vakuumskom mlaznicom (Slika 4) ili hvataljkom te staviti na odgovarajuće mjesto na PCB.

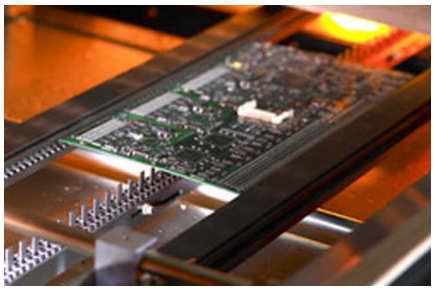
Slika 4. Pokupljanje komponenti vakuumskom mlaznicom [1]

4. AOI prije *reflow* lemljenja

AOI sustav je na ovoj poziciji izuzetno bitan kako bi se potvrdilo da su svi SMD-ovi pravilno pozicionirani. AOI provjerava postojanje i tip komponente te njenu orijentaciju.

5. *Reflow* lemljenje

Reflow lemljenje radi se u *Reflow* pećima u kojima se formiraju spojevi između komponenti i PCB-a, na način da se PCBA dovede na određenu temperaturu (Slika 5). Ovaj proces se može činiti znatno jednostavnijim od ostalih, no kako bi se proces lemljenja pravilno proveo potrebno je odrediti optimalan temperaturni profil koji će osigurati kvalitetan spoj, ali neće oštetiti komponente.



Slika 5. PCBA u *reflow* peći [1]

6. AOI poslije *reflow* lemljenja

Kako bi se osigurao ispravan rad PCBA-a, AOI sustav nakon *reflow* lemljenja treba provjeriti kvalitetu spojeva između komponenti i PCB-a. Većina industrija zahtjeva upravo ovaj način provjere, koji im osigurava valjanost svakog dostavljenog PCBA-a.

7. Selektivno lemljenje

Selektivno lemljenje se često koristi za *through-hole* komponente[6], ali i za komponente osjetljive na visoke temperature koje zbog toga ne smiju prolaziti kroz peć.

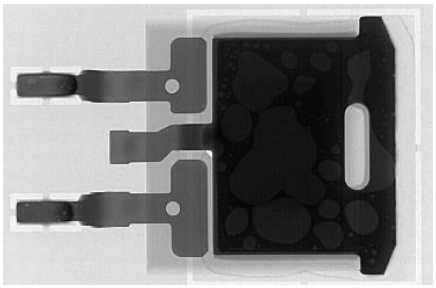
Slika 6. Selektivno lemljenje [1]

Umjesto selektivnog lemljenja može se koristiti valno lemljenje opisano u [8] i [9].

[6] *Trough-hole* komponente su dizajnirane tako da prolaze kroz rupe na PCB-ima, a koriste se zbog veće čvrstoće.

8. RTG inspekcija

RTG inspekcija podrazumijeva otkrivanje tipično skrivenih karakteristika rendgenskim zrakama [11]. U ovom slučaju koristi se za provjeru kvalitete lemnih nožica, čime se osigurava optimalno stanje svih komponenti (Slika 7).



Slika 7. Rezultat RTG inspekcije tranzistora [1]

2.2. AOI tehnologija

AOI je tehnologija strojnoga vida kojoj je cilj zamijeniti manualnu vizualnu inspekciju koja se radi kako bi se provjerila ispravnost i kvaliteta u proizvodnji. Prema [12], ova tehnologija podrazumijeva integraciju optike, mehanizama, elektroničke kontrole i programa u svrhu zamjene ljudskih očiju. Sustav se oslanja na specifični algoritam koji identificira greške proizvoda, a osim greški, mogu se mjeriti i različite karakteristike proizvoda na razini koju ljudsko oko ne može raspoznati.

Krajnji cilj sustava strojnog vida je stvoriti model realnog svijeta iz slika [19]. Općenito sustavi strojnoga vida funkcioniraju na način da se kamerom zabilježi stanje od interesa koje se potom algoritmima računalnog vida procesira i interpretira. Zatim se, s obzirom na rješenje, ostalim komponentama sustava javlja kako da reagiraju.

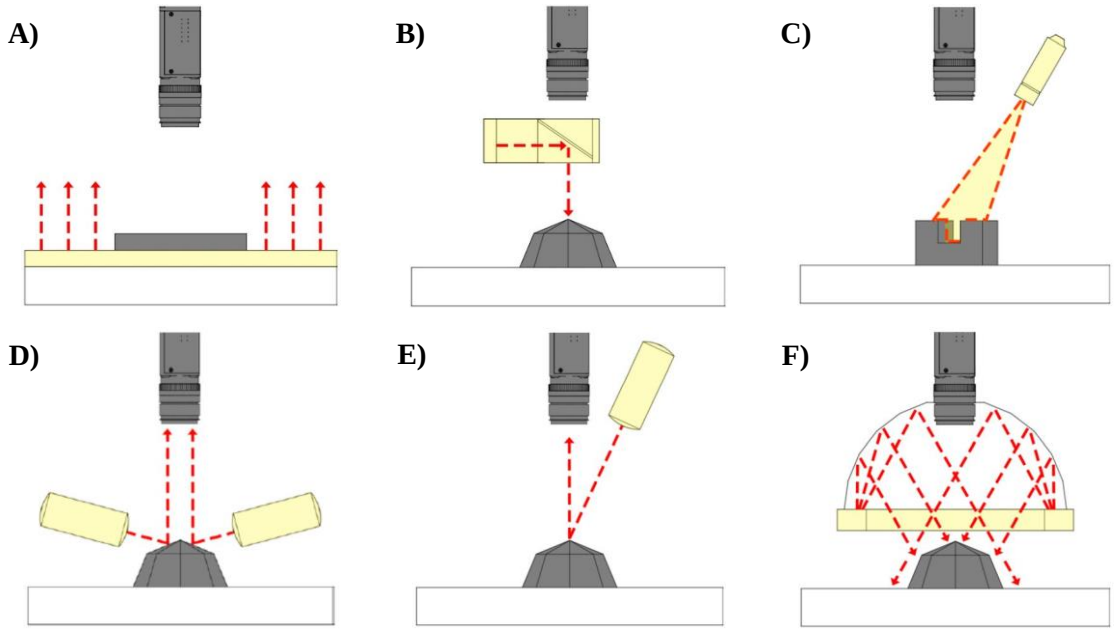
Osnovne komponente strojnog vida podrazumijevaju osvjetljenje, optiku, komunikaciju i procesiranje vida.

2.2.1. Osvjetljenje

Budući da sustavi strojnog vida stvaraju sliku analizirajući reflektirano svjetlo s površine promatranog objekta, jedna od najbitnijih komponenti strojnog vida je osvjetljenje.

Postoje različite tehnike osvjetljenja koje podrazumijevaju izvor svjetlosti i njegovu poziciju u odnosu na promatrani objekt i kameru. Odabrana tehnika osvjetljenja može značajno poboljšati kvalitetu i rezoluciju slike, a razlikuju se dalje navedeni tipovi osvjetljenja.

- Pozadinsko osvjetljenje najčešće se koristi kako bi se pojačao obris objekta u svrhu detekcije njegova ruba ili mjerenja vanjskih dimenzija (Slika 8A).
- Aksijalno difuzno osvjetljenje daje ravnomjerno osvijetljene i homogene slike (Slika 8B).
- Strukturirano osvjetljenje daje površinske preglede neovisne o kontrastu površine koji su potrebni za dobivanje dimenzijskih podataka i izračunavanje volumena (Slika 8C).
- Osvjetljenje pod malim kutom koristi se za detekciju greški na površinama s niskim kontrastom (tamne površine) (Slika 8D).
- Osvjetljenje pod velikim kutom koristi se za detekciju greški na površinama s visokim kontrastom (svijetle površine) (Slika 8E).
- Difuzno kupolno osvjetljenje daje najujednačenije osvjetljenje značajki od interesa i može zamaskirati nepravilnosti koje ne ulaze u zonu interesa (Slika 8F).



Slika 8. Različiti tipovi osvjetljenja: A) pozadinsko, B) aksijalno difuzno, C) strukturirano, D) pod malim kutom, E) pod velikim kutom, F) difuzno kupolno [15].

Tehnika osvjetljenja koja će se koristiti ovisi o tome kako se svjetlo reflektira od površine od interesa, pri čemu je cilj dobiti sliku što većeg intenziteta. Intenzitet slike se naziva i ozračenje slike (engl. *image irradiance*). Ozračenje jedne točke slike se definira kao snaga po jedinici površine zračenja koja pada na ravninu slike. Ozračenje u točki slike $E(x', y')$ se definira kao količina energije koja zrači u odgovarajučoj točki u sceni $L(x, y, z)$ u smjeru točke slike:

$$E(x', y') = L(x, y, z). \quad 2.1$$

Kada se uzmu u obzir gubici koji se događaju zbog različitog faktora refleksije površine i kuta upada zrake dobiva se bidirekcionalna funkcija distribucije reflektivnosti:

$$E(\theta_i, \phi_i) = \frac{L(\theta_e, \phi_e)}{f(\theta_i, \phi_i, \theta_e, \phi_e)}, \qquad 2.2$$

gdje je: (θ_i, ϕ_i) – smjer u polarnim koordinatama relativno s obzirom na površinu,

(θ_e, ϕ_e) – smjer u kojem se energija emitira s površine.

Poznavajući jednadžbu 2.2 može se odrediti potrebno osvjetljenje, iz čega slijedi generalna jednadžba za računanje ukupnog ozračenja površine iz generalnog izvora svjetlosti:

$$I_0 = \int_0^{2\pi} \int_0^{\pi/2} I(\theta_i, \phi_i) \sin\theta_i \cos\theta_i \, d\theta_i d\phi_i, \qquad 2.3$$

gdje je $I(\theta_i, \phi_i)$ zračenje po jedinici čvrstog kuta prolazeći kroz hemisferu iz smjera (θ_i, ϕ_i).

S obzirom na vrstu površine na koju pada zračenje razlikuju se površine koje upijaju, reflektiraju, propuštaju, rasipaju ili emitiraju svjetlost.

2.2.2. Optika

Sustavi strojnog vida se zasnivaju na *pinhole* modelu kamere [13], gdje se u praksi leća postavlja na takav model kako bi se skupilo više svjetlosti na senzor. Navedeno omogućava rad sustava u uvjetima lošijeg osvjetljenja ili mogućnost rada s većom brzinom zatvarača, no ograničava dubinu vidnog polja. Kako bi se odabrala adekvatna leća potrebno je izračunati dubinu vidnog polja. Jednadžba leće je sljedeća:

$$\frac{1}{d'} - \frac{1}{d} = \frac{1}{f}, \qquad 2.4$$

gdje: d' – predstavlja udaljenost ravnine slike od optičkog ishodišta kamere,

d – predstavlja udaljenost do točke u sceni,

f – predstavlja fokalnu duljinu leće.

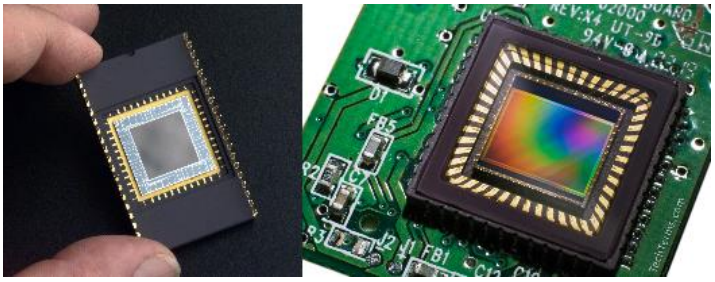
Kako bi se što bolje razlikovale bitne karakteristike na slici potrebno je imati dobru rezoluciju. Rezolucija slike ovisi o udaljenosti piksela, aberaciji leće, difrakciji i dubini polja, no za većinu sustava strojnoga vida i distorzija leće i difrakcija imaju mali utjecaj. Rezoluciju slike najviše limitira udaljenost piksela unutar fotosenzibilnog senzora, tj. udaljenosti kristala fotografskog filma, pri čemu je tipična udaljenost 5 μm. Kristalima fotografskog filma

mijenjaju se karakteristike ovisno o tome koliko svjetlosti primaju. Povećanjem veličine kristala povećava se i osjetljivost na svjetlo, što za posljedicu ima smanjenje rezolucije.

Fotosenzitivni senzori koji se najčešće nalaze u kamerama dijele se u dvije grupe [14]:

1. Senzori bazirani na fotoemisijskom efektu koji iskorištavaju fotoelektrični efekt[7],
2. Senzori bazirani na fotovoltaičnom efektu[8], kao što su diode ili fotootpornici.

U kamerama se najčešće koristi CCD ili CMOS semi-kondukcijski fotootporni senzor. Iako su oba razvijena 1960-ih i 1970-ih godina, CCD tehnologija je sazrjela prije i počela se koristiti 1970-ih, dok se CMOS tehnologija počela jače razvijati tek 1990-ih.



Slika 9. CCD senzor (lijevo) i CMOS (desno)

2.2.3. Komunikacija

Svi elementi sustava strojnog vida se moraju moći lako i brzo spojiti pošto se za sklapanje sustava često koriste gotove komponente. To se tipično radi komunikacijom s vanjskim sustavima preko diskretnih I/O točaka (prekidači, PLC-i itd.) ili preko slanja podataka pomoću protokola (RS-232, Ethernet, EtherNet/IP itd.).

2.2.4. Procesiranje vida

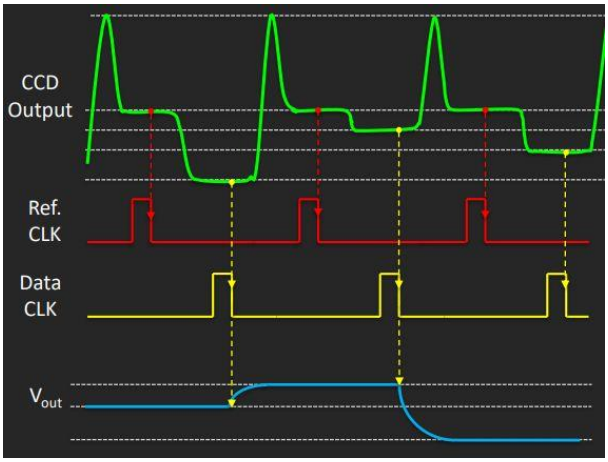
Procesiranje vida je mehanizam izuzimanja informacija iz digitalne slike, a može se odvijati eksterno u sustavu s računalom ili interno u samostojećem vizijskom sustavu. Procesiranje se sastoji od više koraka:

1. Dobivanje slike sa senzora:

Svjetlost se pretvara u naboj pomoću fotosenzibilnog senzora (fotootpornika), potom se naboj pretvara u napon i pojačanje, nakon čega se iz napona izdvaja korisni signal od

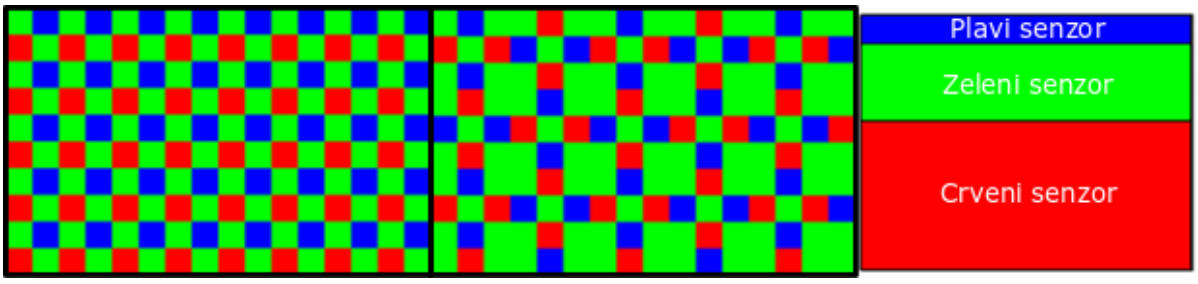
[7] Fotoelektrični efekt se javlja prilikom djelovanja elektromagnetnog zračenja dovoljno kratke valne duljine koje dovodi do izbijanja elektrona iz obasjanog materijala. [16]
[8] Fotovoltaični efekt – proces koji generira napon ili struju unutar fotovoltaične čelije kada je izložen suncu. [18]

referentnog nivoa napona. Prebrojavaju se elektroni nastali izlaganjem svjetlosti te se potiskuje šum nastao fluktuacijom signala nakon resetiranja fotootpornika. Nad korisnim signalom se radi korelirano dvostruko uzorkovanje[9] (CDS, engl. *correlated double sampling*).



Slika 10. CDS CCD senzora [17]

Slika u boji se dobiva rastavljanjem spektra na tri komponente: crvenu (engl. *Red*), zelenu (engl. *Green*) i plavu (engl. *Blue*) (Slika 11). S obzirom da sensel[10] senzora nije kromatski osjetljiv, koriste se filtri kako bi bio osjetljiv na jednu od boja. Dobivanje cijelog spektra boja na slici se radi s tri senzora koja imaju tri različita filtera. Postoji više načina na koje se navedeno može napraviti: Bayerova matrica[11], Foveon senzor[12] i X-Trans senzor[13]. Svaka ima svoje prednosti i mane, a trenutno je Bayerova matrica, iako najstarija metoda, najrasprostranjenija.



Slika 11. Bayerova matrica (lijevo), X-Trans matrica (sredina) i Foveon senzor (desno)

[9] Korelirano dvostruko uzorkovanje je metoda za mjerenje električnih vrijednosti poput napona ili struja koji omogućuje uklanjanje neželjenog pomaka.
[10] Sensel je osnovni jedinični senzorski element optičkog senzora, četiri susjedna čine jedan piksel.
[11] Bayerova matrica ima tri filtera koji reprezentiraju pojedini sensel, četvrti sensel hvata zelenu boju.
[12] Foveon senzor koristi tri vertikalno naslagane fotodiode koje imaju različitu spektralnu osjetljivost.
[13] X-trans senzor sličan je Bayerovoj matrici, ali su senseli u znatno većem udjelu predstavljeni zelenom bojom.

2. Predprocesiranje slike:

Pojam podrazumijeva sve operacije koje se obavljaju nad slikom prije nego se krene procesirati. Predprocesiranje slike radi se različitim filterima u prostornoj ili frekvencijskoj domeni. Razlog tomu je poboljšanje kvalitete slike. Prema [13] najvažnije metode filtriranja su:

- Modifikacija histogramom

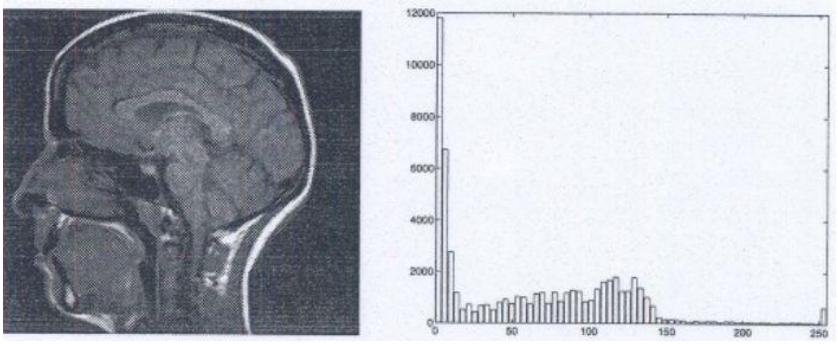
Modifikacija histogramom je metoda za rastezanje kontrasta slika koje sadrže neravnomjerno raspoređene nijanse sive, pri čemu se karakteristike od interesa nalaze u malom spektru vrijednosti nijansi (Slika 12).

Slika 12. Slika s lošim kontrastom [13]

Primjer metode je skaliranje slike gdje su $[a, b]$ raspon piksela slike koji se žele ekspandirati na raspon $[c_1, c_k]$. Jednadžba za mapiranje originalne vrijednosti piksela c iz originalnog raspona u vrijednost piksela c' u novom rasponu je:

$$c' = \frac{c_k - c_1}{b - a}(c - a) + c_1. \qquad 2.5$$

Problem ove sheme je da nastali histogram ima razmake između sekcija (Slika 13).



Slika 13. Slika nakon skaliranja histogramom [13]

Jedna od boljih, ali malo kompleksnijih, metoda može se koristiti ako se prethodno zna distribucija sivih od interesa, a prikazana je jednadžbom:

$$\sum_{i=1}^{k_1-1} p_i \leq q_1 < \sum_{i=1}^{k_1} p_i, \qquad 2.6$$

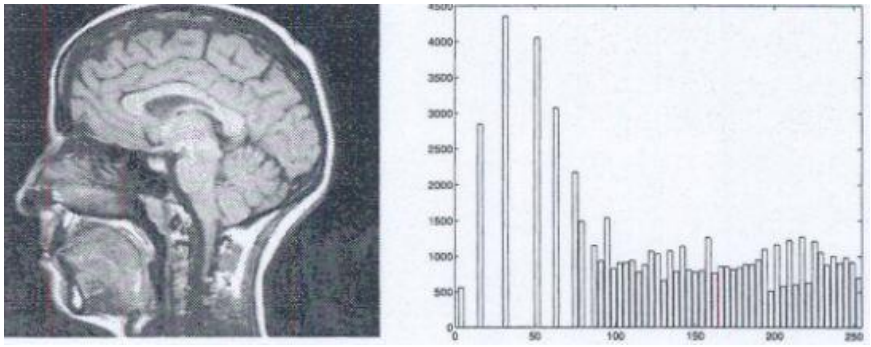
gdje su: p_i – broj piksela na razini z_i u originalnom histogramu,

q_i – broj piksela na razini z_i u željenom histogramu.

Dalje se traži k_2 vrijednost tako da vrijedi:

$$\sum_{i=1}^{k_2-1} p_i \leq q_1 + q_2 < \sum_{i=1}^{k_2} p_i, \qquad 2.7$$

pri čemu se postupak odvija sve dok sve vrijednosti sive iz originalnog histograma nisu obrađene. Krajnji histogram u tom slučaju izgleda kao na Slika 14.



Slika 14. Slika nakon izjednačavanja histograma [13]

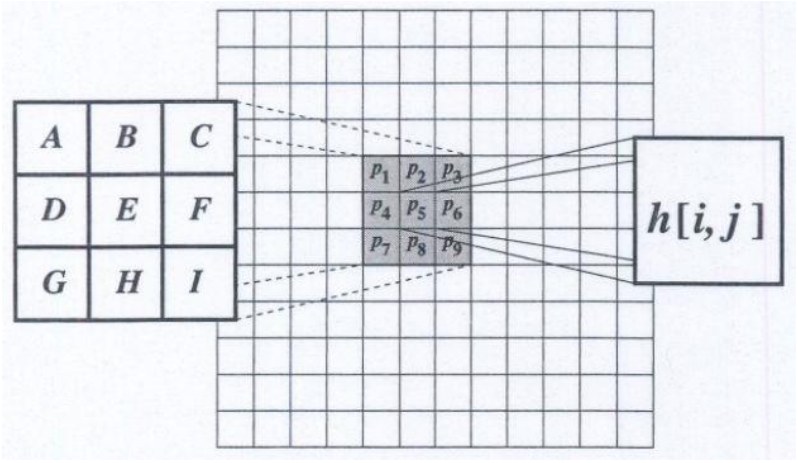
- Linearni sustavi:

Većina operacija procesiranja slike se može modelirati kao linearni sustavi. Iz toga slijedi da je izlaz $h(x,y)$ jednak konvoluciji između ulazne slike $f(x,y)$ i filtra $g(x,y)$ prema:

$$h(x,y) = f(x,y) * g(x,y) = \iint_{-\infty}^{\infty} f(x',y')g(x-x',y-y')dx'dy'. \qquad 2.8$$

Za diskretne funkcije vrijedi:

$$h[i,j] = f[x,y] * g[x,y] = \sum_{k=1}^{n}\sum_{l=1}^{m} f[k,l]g[i-k,j-l]\,. \qquad 2.9$$



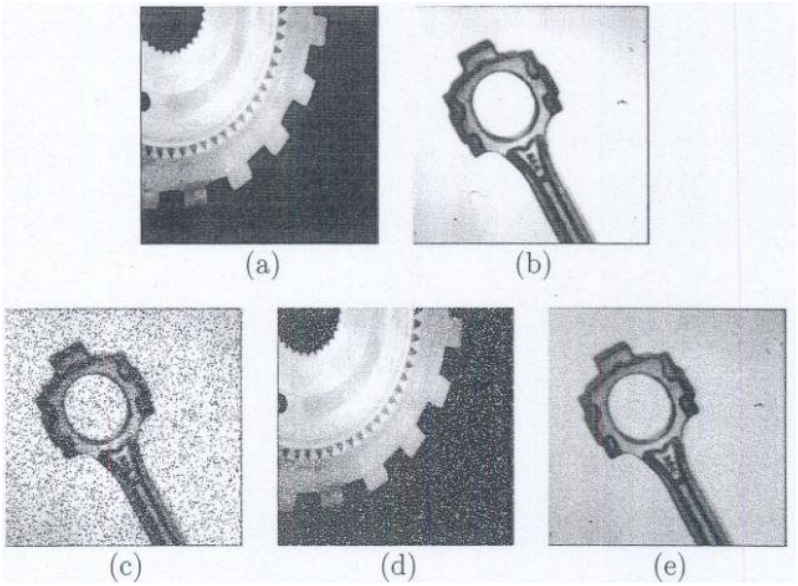
Slika 15. Konvolucijska maska

Sljedeća jednadžba prikazuje konvoluciju sa Slika 15:

$$h[i,j] = Ap_1 + Bp_2 + Cp_3 + Dp_4 + Ep_5 + Fp_6 + Gp_7 + Hp_8 + Ip_9 \qquad 2.10$$

- Linearni filtri

Kao što je i prethodno spomenuto, slike su često iskvarene kada se pokupe sa senzora. Tipični primjeri buke koja uzrokuje iskvarenost slike su: buka soli i papra (engl. *salt and pepper noise*), impulsna buka i Gaussianova buka.

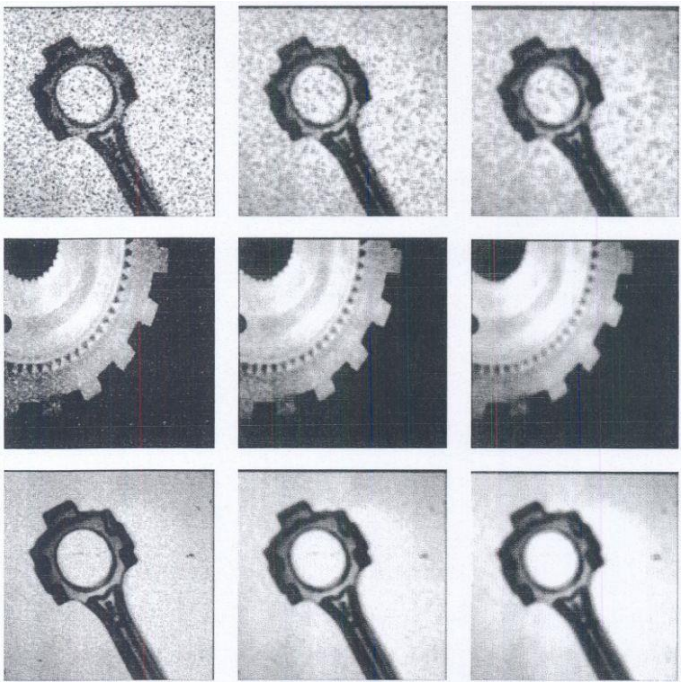


Slika 16. Prikaz slika iskrivljenih bukom: (a) i (b) originalne slike, (c) buka soli i papra, (d) impulsna buka i (e) Gaussianova buka.

Linearni filteri su vrsta filtera koji za rezultat daju ponderirani zbroj piksela, a najpoznatiji i najjednostavniji je Mean filter. Kod Mean filtera svaki piksel je zamijenjen prosjekom vrijednosti svih piksela u lokalnom susjedstvu na način prema jednadžbi:

$$h[i,j] = \frac{1}{M} \sum_{(k,l)\in N} f[k,l], \qquad 2.11$$

gdje M označava ukupni broj piksela u susjedstvu veličine N te površinu filtera, tj. kernela.

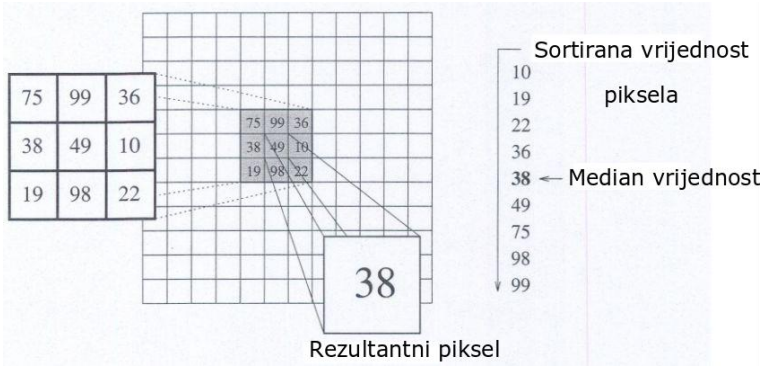


Slika 17. Buka sa Slika 16 filtrirana Mean filterom, veličine kernela s lijeva na desno: 3x3, 5x5 i 7x7

Korištenjem ovog filtera buka se uspješno smanjuje no gube se oštri detalji u slici.

- Median filter

Ovaj filter radi na sličan način kao i Mean filter, ali umjesto da radi prosjek svih susjednih piksela slaže sve piksele po veličini i odabire onaj čija se vrijednost nalazi u sredini. Proces je prikazan na Slika 18.



Slika 18. Rad Median filtera [13]

- Zaglađivanje Gaussianom

Gaussian filteri su klasa linearnih filtera za zaglađivanje s težinama zaglađivanja odabranih prema Gaussianovoj jednadžbi:

$$g(x) = e^{-\frac{x^2}{2\sigma^2}}, \tag{2.12}$$

koja za svrhe procesiranja slike poprima oblik:

$$g[i,j] = e^{-\frac{i^2+j^2}{2\sigma^2}}, \tag{2.13}$$

gdje je σ širina Gaussiana.

U suštini, susjedni pikseli najbliži ciljanom pikselu će imati najveći utjecaj na njegovu krajnju vrijednost. Ova vrsta filtera zbog svojih različitih svojstva je efektivni nisko-propusni filter iz perspektive prostorne i frekvencijske domene. Filteri se lako implementiraju i mogu se koristiti u praktičnoj primjeni u vizijskim aplikacijama.

3. Procesiranje i donošenje odluke:

Procesiranje slike se zasniva na različitim tehnikama kojima se traže ili ističu informacije sa slike od interesa. Procesiranje podrazumijeva: detekciju rubova, segmentaciju i strojno učenje.

- Detekcija ruba:

Detekcija rubova je često prvi korak prilikom rađenja analize slika jer rubovi sadrže najviše semantičkih informacija. Rubovi sami po sebi reprezentiraju nagle promjene u intenzitetu na slici, tj. skokove intenziteta tako da je cilj detekcije rubova odrediti mjesta gdje dolazi do tih skokova. U jednoj dimenziji rub je povezan s lokalnim vrhom u prvoj derivaciji, gdje gradijent određuje promjenu funkcije, pri čemu se slika može definirati kao matrica niza uzoraka nekih kontinuiranih funkcija intenziteta slike [13]. Bitne su dvije karakteristike povezane s gradijentom, a to su snaga i smjer gradijenta, pri čemu snaga gradijenta označava snagu ruba, a smjer gradijenta lokalnu orijentaciju ruba. Za digitalne slike se često koristi aproksimacija gradijenta gdje je najjednostavnija aproksimacija gradijenta prikazana jednadžbom:

$$G_x \cong f[i,j+1] - f[i,j], \tag{2.14}$$

$$G_y \cong f[i,j] - f[i+1,j], \tag{2.15}$$

za j u smjeru x, te i u smjeru y. Navedeno se može implementirati u jednostavne konvolucijske 2 x 2 matrice prikazane u 2.16.

$$G_x = \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix} \; G_y = \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix} \qquad 2.16$$

Prema prethodnom prikazanom, detektirana točka će se nalaziti između četiri piksela 2 x 2 matrice, stoga se u praksi koriste 3 x 3 konvolucijske matrice gdje se gradijent računa oko piksela u centru.

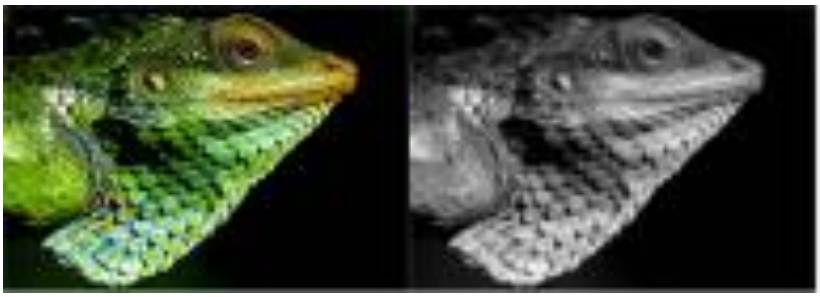
Algoritmi za detekciju ruba mogu sadržavati jedan ili više od sljedeća tri koraka: filtriranje buke, pojačanje piksela gdje je značajna promjena lokalnog intenziteta te detekcija. Jedan od poznatijih algoritama ovog tipa je Sobelov filter, koji ima sljedeću formu:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \; G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \qquad 2.17$$

Detektori rubova dobiveni prvom derivacijom za nuspojavu imaju detekciju previše točaka rubova, tj. umjesto tankih linija dobiju se debele. Bolji pristup bi svakako bio da se nađu točke koje imaju samo lokalni maksimum gradijenta. Uz metodu detekcije rubova pomoću prve derivacije funkcije postoji i metoda detekcije rubova pomoću druge derivacije funkcije, još zvana i detekcija ruba bazirana na prolasku kroz nultu točku (engl. *zero crossing*). Metode koje koriste drugu derivaciju se rijetko koriste u vizijskim sustavima jer su jako osjetljive na buku stoga ovdje neće biti obrađene.

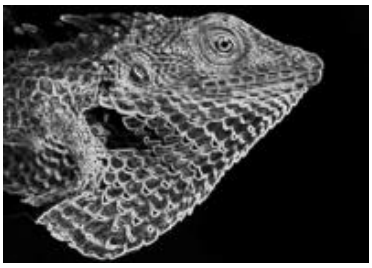
Kako bi se smanjile nuspojave koje nastaju korištenjem prve derivacije razvijen je jedan od najraširenije korištenih filtera za detekciju ruba, Canny edge detektor. Proces ovog algoritma se odvija u sljedećim koracima:

a. Primjenjivanje Gaussian filtera u svrhu zaglađivanja slike te smanjenja buke.



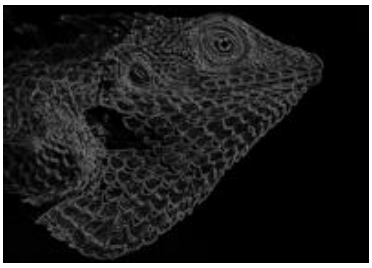
Slika 19. 1. korak *Canny edge* detektora: originalna slika (lijevo) i *Gray scale* slika nakon 1. koraka (desno)

b. Korištenje prve derivacije za detekciju rubova.



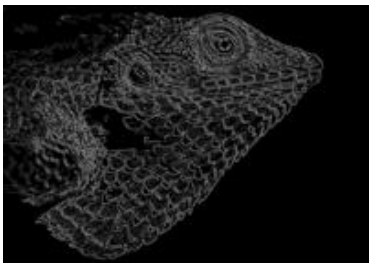
Slika 20. 2. korak *Canny edge* detektora

c. Primjena *non-maximum suppression* metode koja filtrira jedan element od više prisutnih. To se radi tako da filter prolazi kroz sliku i ostavlja vrijednosti koje prelaze određeni prag čime se dokidaju tanki rubovi, tj. rubovi nastali zbog šuma.



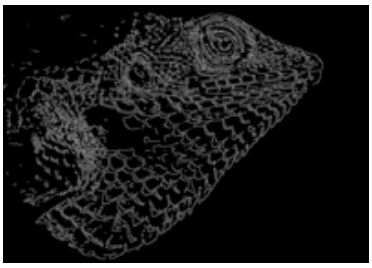
Slika 21. 3. korak *Canny edge* detektora

d. Primjena *double treshold* metode kako bi se pronašli potencijalni rubovi nastali zbog utjecaja buke i boje.



Slika 22. 4. korak *Canny edge* detektora

e. Funkcijom histereze se uklanjaju zaostali slabi rubovi ukoliko nisu povezani s nekim jakim rubom.



Slika 23. 5. korak *Canny edge* detektora

- Segmentacija:

Proces segmentacije se definira kao pronalaženje particije S slike S_p u setu n regija R_i za dan set piksela S_p i mjeru za homogenost (engl. *homogeneity predicate*) $P(\cdot)$ [13], tj. grupiranje piksela slike u regije. Navedeno zapisano jednadžbom izgleda ovako:

$$\bigcup_{i=1}^{n} R_i = S_p. \tag{2.18}$$

Pritom mora vrijediti da mjera za homogenost i particija slike imaju karakteristike da bilo koja regija zadovoljava mjeru:

$$P(R_i) = True, \tag{2.19}$$

tj. da svaka točka particije ima neku zajedničku karakteristiku. Također mora vrijediti da dvije susjedne regije ne mogu biti spojene u istu regiju koja zadovoljava mjeru:

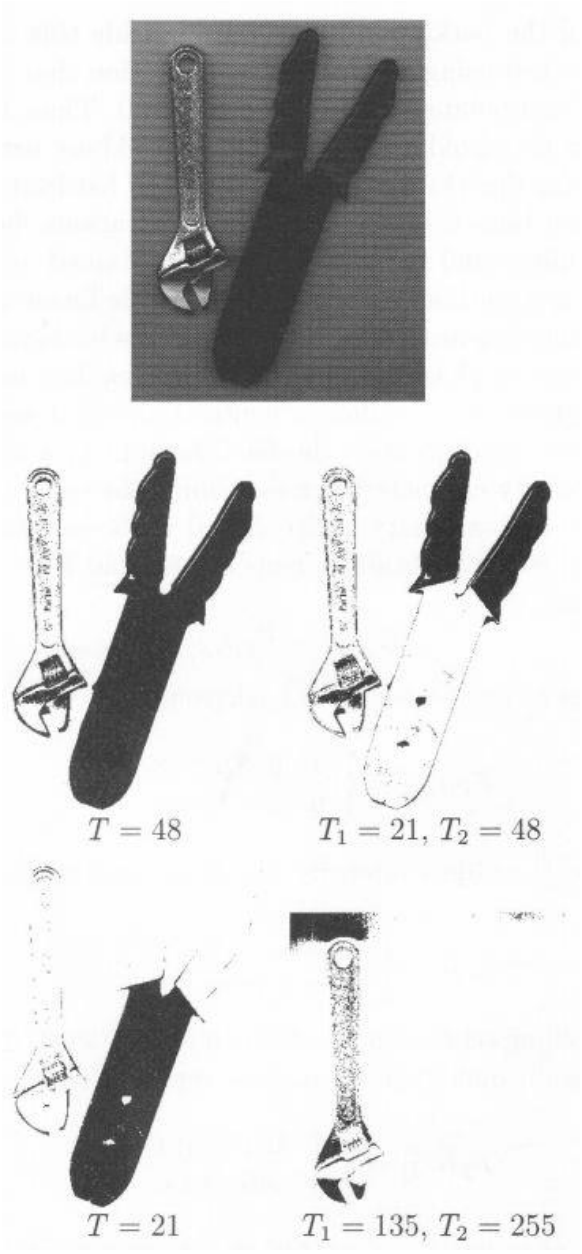
$$P(R_i \cap R_j) = False. \tag{2.20}$$

Najjednostavnija izvedba segmentacije je pragiranje slike (engl. *image thresholding*), gdje se iz slike u sivoj skali (engl. *gray scale image*) dobiva binarna slika prema sljedećoj generalnoj formuli:

$$F_T[i,j] = \begin{cases} 1, & \text{ako je } F[i,j] \in \boldsymbol{Z} \\ 0, & \text{inače} \end{cases}, \tag{2.21}$$

gdje je $\boldsymbol{Z}$ set vrijednosti intenziteta $[T_1, T_2]$ objekta.

Uz navedenu metodu postoje još: metoda podjele histogramom, klasterizacija metodom K-srednjih vrijednosti (engl. *K-means clustering*), metoda pomaka srednjih vrijednosti (engl. *mean shift*), teorijske metode grafova, metode bazirane na ekspertnim sustavima te segmentacija konvolucijskim neuronskim mrežama.



Slika 24. *Image thresholding*:
Originalna slika gore i pragiranje sa različitim pragovima ispod [13].

- Strojno učenje:

Prema [21], strojno učenje se može objasniti kao računalne metode koje, koristeći iskustvo, poboljšavaju performanse ili rade točna predviđanja. Iskustvo se ovdje odnosi na informacije koje su dostupne učeniku, što tipično ima oblik baze podataka u elektroničkom obliku, skupljene i dostupne za analizu. Ova vrsta procesiranja podataka koristi se kod složenih problema, problema s ogromnom količinom podataka, sustava koji se dinamički mijenjaju, itd. Tipovi strojnog učenja su:

a. Nadgledano učenje (engl. *Supervised learning*):

Učenik za treniranje prima set podataka koji sadrže željene izlazne podatke, nakon čega radi predikcije s obzirom na još neviđene podatke. Ovdje se radi o najčešće korištenom tipu strojnog učenja koji će biti korišten i unutar ovoga rada u formatu konvolucijskih neuronskih mreža.

b. Nenadgledano učenje (engl. *Unsupervised learning*):

Učenik za trening prima set podataka koji ne sadrže željene izlazne podatke, nakon čega radi predikcije za neviđene podatke. Primjeri ovog tipa su klasterizacija i dimenzionalna redukcija.

c. Polunadgledano učenje (engl. *Semi-supervised learning*):

Učenik za trening prima set podataka od kojih neki sadrže željene izlazne vrijednosti dok drugi ne, nakon čega radi predikcije za neviđene podatke. Koristi se kada je lako skupiti podatke, ali ih je skupo označiti/kategorizirati.

d. Transduktivno zaključivanje (engl. *Transductive inference*):

Kao i kod polunadgledanog učenja, učenik prima set podataka od kojih samo neki sadrže željene izlazne vrijednosti, no u ovom slučaju je cilj predvidjeti samo testne podatke iz seta podataka koji nisu imali izlazne vrijednosti.

e. On-line učenje:

Za razliku od prethodno navedenih tipova, ovaj tip prima podatke jedan po jedan pri čemu svaki puta radi estimaciju rješenja te validaciju točnosti rješenja (prima podatke bez pripadajućih izlaznih vrijednosti). Ukoliko je predikcija bila kriva, radi se korekcija. Nakon korekcije dolazi novi podatak i postupak se ponavlja.

f. Podržano/ojačano učenje (engl. *Reinforcement learning*):

U ovom slučaju učenik djeluje samostalno u okolini te s obzirom na poduzete sekvence akcija dodjeljuju mu se nagrade ili kazne. Cilj učenika je maksimizirati dobivene nagrade kroz slijed akcija i interakcija s okolišem.

g. Aktivno učenje (engl. *Active learning*):

Učenik adaptivno ili interaktivno skuplja podatke za trening, tipično postavljanjem upita „proroku“ (engl. *Oracle*) kako bi pristupio kategorizaciji za promatrane podatke. Ovaj tip učenja se koristi u aplikacijama gdje je teško napraviti kategorizaciju, kao na primjer u aplikacijama za računalnu biologiju.

Uz navedene, u praksi postoji još niz različitih i kompleksnijih scenarija strojnog učenja. Ovdje su navedeni osnovni, koji se i najčešće koriste.

Iz svega navedenoga moguće je vidjeti kako je za razvoj AOI algoritma potrebno paziti na veliku količinu parametara koji moraju biti što konzistentniji (osvjetljenje, parametri promatrane površine, okolni uvjeti...) kako bi analiza slike bila što točnija. No u slučaju da se ti parametri ne mogu kontrolirati te ako su oni varijabilni, mogu se koristiti tehnike strojnoga vida kako bi se postigao isti rezultat. Iz toga razloga će, unutar ovoga rada, biti korišten tip strojnog učenja, nadgledano učenje, tj. konvolucijske neuronske mreže.

2.3. Konvolucijske neuronske mreže

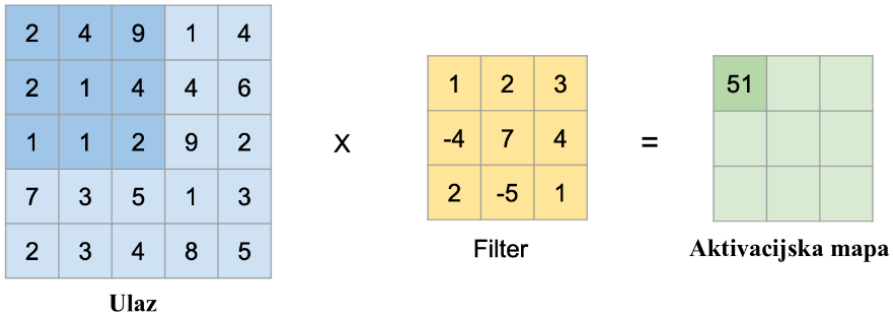
Konvolucijska neuronska mreža (CNN, engl. *convolutional neural network*) je tip neuronske mreže[14] koja iskorištava činjenicu da joj je ulaz slika što dozvoljava programiranje određenih karakteristika u arhitekturu čime se unaprijedni (engl. *feed forward*) algoritam[15] čini učinkovitijim te se smanjuje broj potrebnih parametara u mreži. Ime dolazi od matematičke linearne operacije konvolucije prethodno spomenute u 2.2.4. Slojevi algoritma sadrže neurone posložene u tri dimenzije: širinu, visinu i dubinu, što je potrebno jer na ulaz mreže može doći slika u boji koja ima dimenzije širina x visina x 3 (dubina). To se može uočiti i ako se pažljivije prouči 2.2.4 gdje se objašnjava kako se preuzimaju podaci sa senzora.

Arhitektura CNN-a se sastoji od različito posloženih slojeva: konvolucijski sloj (CONV, engl. *convolutional layer*), nelinearni sloj (engl. *nonlinearity layer*), sloj sažimanja (engl. *pooling*) i potpuno povezani sloj (FC, engl. *fully-connected layer*).

2.3.1. CONV sloj

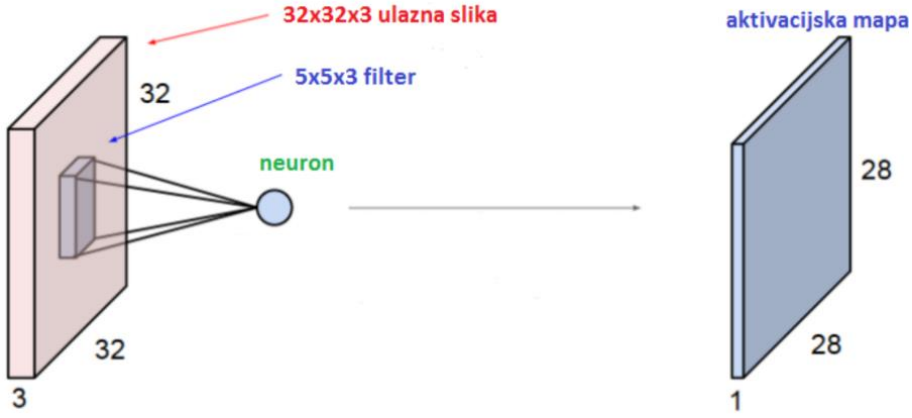
CONV sloj je jezgra CNN-a koja obavlja većinu računanja. Poanta CONV sloja je brzo učenje velikih setova filtera u paraleli kako bi se riješili specifični problemi kao klasifikacija slike. Cijeli sloj je baziran na matematičkoj funkciji konvolucije prethodno obrađene u tekstu. Slika 25 prikazuje način rada konvolucijskog sloja na primjeru jednog filtera.

[14] Neuronska mreža je sustav za računanje sastavljen od jednostavnih elemenata koji se zovu neuroni, a bazira se na strukturi i načinu na koji ljudski mozak procesuira informacije. Više u [22].
[15] Unaprijedni algoritam je tip algoritma neuronske mreže gdje je svaki element sloja povezan sa svakim elementom u prethodnom sloju.



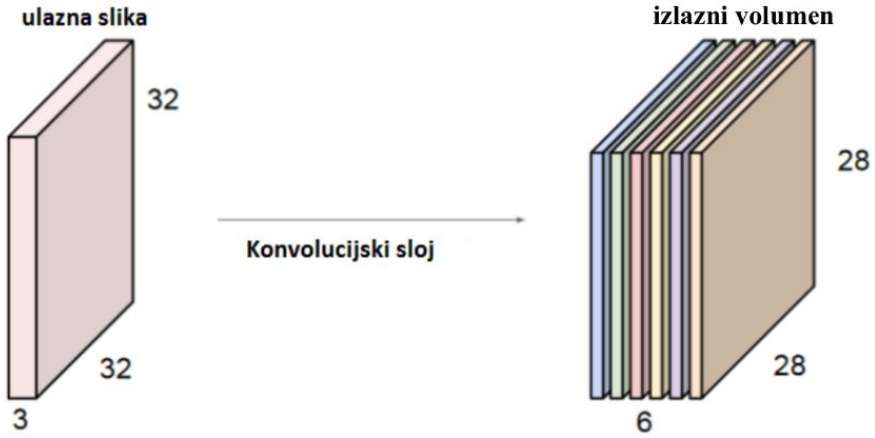
Slika 25. Rad filtera

Svakim prolaskom filtera preko slike stvara se jedna 2-D aktivacijska mapa koja daje reakcije filtra za svaku poziciju na kojoj se on nalazio, tj. svako polje aktivacijske mape predstavlja jedan skriveni neuron u sloju (Slika 26).



Slika 26. Rad konvolucijskog sloja

Svaki konvolucijski sloj raspolaže s predodređenim brojem filtera koji svojim prolazom čine skupinu aktivacijskih mapa te daju izlazni volumen (Slika 27).



Slika 27. Izlazni volumen

2.3.2. Nelinearni sloj

Nelinearni sloj se koristi za podešavanje ili odsijecanje generiranog izlaza. To se radi kako bi se izlaz zasitio ili ograničio. Godinama su se koristile sigmoidna i tangens hiperbolna funkcija za neliearnost, no u zadnje vrijeme je jako popularna primjena ReLU (engl. *Recitified Linear Unit*) funkcije, i to iz sljedećih razloga:

1. ReLU ima jednostavniju definiciju funkcije i gradijenta što je moguće vidjeti u jednadžbama 2.22 i 2.23.

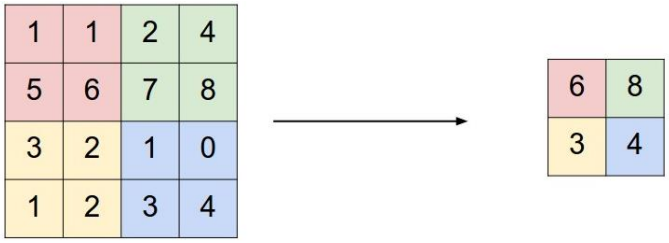
$$ReLU(x) = \max(0, x) \tag{2.22}$$

$$\frac{d}{dx} ReLU(x) = \begin{cases} 1, & ako\ je\ x > 0; \\ 0, & inače. \end{cases} \tag{2.23}$$

2. Zasićene funkcije kao sigmoidna i tangens hiperbolna stvaraju probleme prilikom *backpropagation* procesa (poglavlje 2.3.5, stranica 24). Čim je dublji dizajn CNN-a tim gradijentni signal počinje nestajati („*vanishing gradient*“). To se događa jer je gradijent tih funkcija, svugdje osim u centru, jako blizu nuli. S druge strane, ReLU ima konstantni gradijent za sve pozitivne ulaze.
3. ReLU stvara rjeđi prikaz jer nula u gradijentu dovodi do dobivanja potpune nule, dok sigmoidna i tangens hiperbolna funkcija imaju ne-nula rezultate iz takvog gradijenta što može negativno utjecati na trening.

2.3.3. Sloj sažimanja

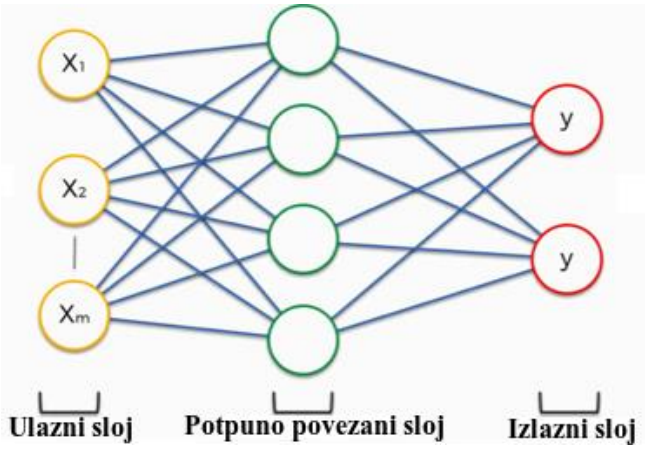
Glavna ideja sloja sažimanja je uzorkovanje prema dolje (engl. *down-sampling*) kako bi se smanjila složenost daljnjih slojeva, nešto slično smanjenju rezolucije slike. Najpopularniji primjer je *max-pooling.* Ima kernel veličine 2×2 i filtrira aktivacijsku mapu iz prethodnog sloja tako da se odabere maksimalni od četiri. Kernel ima pomak 2 tako da nikada ne prelazi preko područja na kojem je bio, tj. odbacuje 75% podataka.



Slika 28. Rezultat *max-pool* sloja

2.3.4. FC sloj

FC sloj ima sličnu formu kao i neuroni raspoređeni u tradicionalnoj neuronskoj mreži (Slika 29).



Slika 29. FC sloj

Ovaj sloj vrši klasifikaciju slika; a za razliku od ostalih matrično formiranih slojeva, FC sloj formiran je vektorski. Kako bi ovaj sloj što bolje radio, često mu prethodi spljošteni sloj (engl. *Flatten*) koji prethodni aktivacijski volumen, koji je izlaz ostalih slojeva, pretvara u vektor. Izlazni sloj FC funkcije se provodi kroz *Softmax* funkciju koja izlazne parametre svodi na vjerojatnosti (Slika 30).

Slika 30. *Softmax* funkcija

2.3.5. Backpropagation

Backpropagation je proces na kojem se zasnivaju neuronske mreže, a funkcionira u sljedećim koracima:

1. Prolaz unaprijed (engl. *forward pass*) funkcionira na način da se s originalnom slikom prođe kroz strukturu mreže. Početne vrijednosti težinskih faktora filtera se biraju slučajnim odabirom.
2. Funkcija gubitaka (engl. *loss function*) radi na način da uspoređuje podatke za učenje s dobivenim vrijednostima za što se najčešće koristi metoda srednje kvadratne pogreške (MSE, engl. *mean squared error*), prema jednadžbi:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2, \qquad 2.24$$

pri čemu je Y_i podatak iz baze za trening, a $\hat{Y}_i$ predviđena vrijednost.

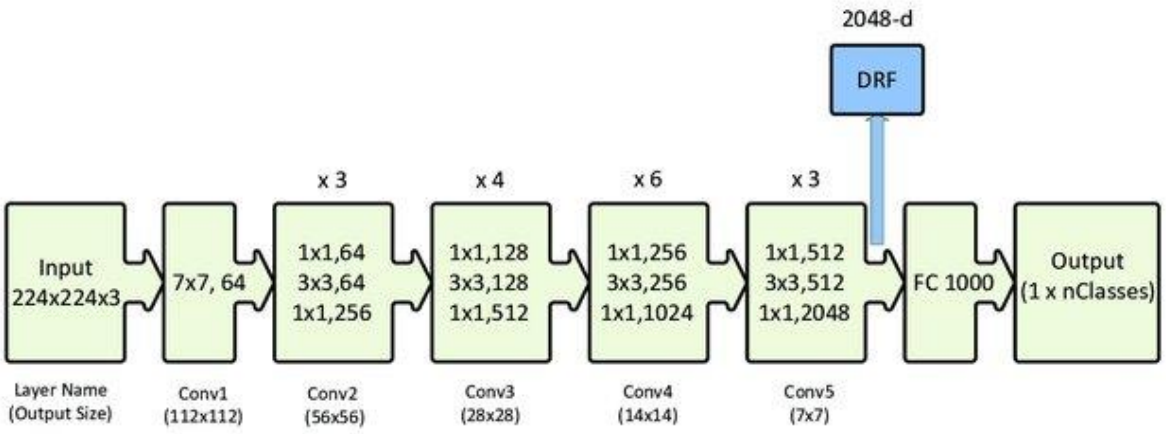
3. Prolaz unatrag (engl. *backward pass*) pronalazi težine koje najviše doprinose stvaranju greške, nakon čega se iste korigiraju metodom stohastičkog gradijentnog spusta (SGD, engl. *stohastic gradient descent*). Cilj je minimizirati funkciju koja ima formu sume prema jednadžbi:

$$Q(\omega) = \frac{1}{n}\sum_{i=1}^{n} Q_i(\omega), \qquad 2.25$$

gdje se procjenjuje parametar ω koji minimizira $Q(\omega)$.

4. Osvježavanje težinskih faktora (engl. *weight update*) uzima sve težinske faktore te ih korigira tako da pridodaju smanjenju greške.

Jedna od poznatijih arhitektura za razvoj AOI algoritma, koja se koristi i unutar ovog diplomskog rada, je ResNet50. Ona sadrži pedeset slojeva, a njena forma prikazana je na sljedećoj slici:



Slika 31. Arhitektura CNN-a ResNet50

Bilo koji CNN s puno slojeva, pa tako i ResNet50, radi na način da se u početnim slojevima prepoznaju najjednostavniji elementi, a to su rubovi. U daljnjim slojevima se ti rubovi slažu i stvaraju oblike, a oblici cjeline. Svaka cjelina može predstavljati neko apstraktno značenje koje više nije jednostavno, kao lice, ramena, krošnje drveta, itd.

3. RAZRADA ZADATKA

U sklopu ovog diplomskog rada osmislila se proizvodna linija za razvoj tiskanih pločica i identificirale su se greške i problemi koji se mogu pojaviti prilikom rada takve linije. Naposljetku, dodatkom sustava na bazi AOI tehnologije te su se greške prepoznale i/ili uklonile.

3.1. Proizvodna linija

Osmišljena je SMT linija koja je 3D modelirana i projektirana u programu *Fusion 360*[16]. Početno je linija modelirana u svrhu provjere uklapanja u poslovni prostor te se prostor adekvatno pripremio kako bi sve bilo spremno za puštanje u pogon. Priprema prostora podrazumijeva dovođenje zraka i struje do sekcija na kojima će se isti spajati na pojedini element linije, postavljanje otvora za odvođenje para koje proizvodi peć te dizajniranje okolnog prostora. Osmišljena linija se sastoji od sljedećih elemenata: sustava za printanje lemne paste, dva *pick and place* stroja, peći za *reflow* tehniku lemljenja te adekvatnih transportera (engl. *conveyor*) za dostavljanje, transport i odlaganje PCB panela. Svi elementi su odabrani iz ponude firme Amtest kojima se jedno od sjedišta nalazi u Hrvatskoj, u gradu Zagrebu. Razlog ovom odabiru je mogućnost brzih intervencija u slučaju zastoja ili kvara te općenito dostupna korisnička podrška.

3.1.1. Sustav za printanje lemne paste

Odabrani sustav za printanje lemne paste je Ekra serio 4000 (Slika 32). Radi se o potpuno automatiziranom sustavu s intuitivnim sučeljem za programiranje.

Slika 32. EKRA serio 4000, render 3D modela

[16] *Fusion 360* – program za 3D modeliranje, simuliranje, renderiranje i proizvodnju.

Tablica 1. Karakteristike EKRA serio 4000 sustava [24]

Ponovljivost poravnanja:	$\pm$12,5µm @ 6 Sigma
Ponovljivost ispisa:	$\pm$25µm @ 6 Sigma
EVA^{TM} – EKRA vizijski sustav za poravnanje	
Vrijeme ciklusa od $11 - 7\ s$ ovisno o opciji	
Brzina mijenjanja između programa < 2 min	
QUEES *squeegee* brzo-izmjenjujući sustav	
Lako rukovanje zahvaljujući SIMPLEX korisničkom sučelju	

Programiranje i postavljanje ovog sustava se radi *offline*.

3.1.2. Pick and place stroj

Odabrani *pick and place* sustav za ovu primjenu je Hanwha SM482 Plus (Slika 33). Radi se o fleksibilnom i pametnom sustavu za postavljanje komponenti od 0603 mikročipova do 22 mm IC komponenti.

Slika 33. Hanwha SM482 Plus, render 3D modela

Tablica 2. Karakteristike Hanwha SM482 Plus [25]

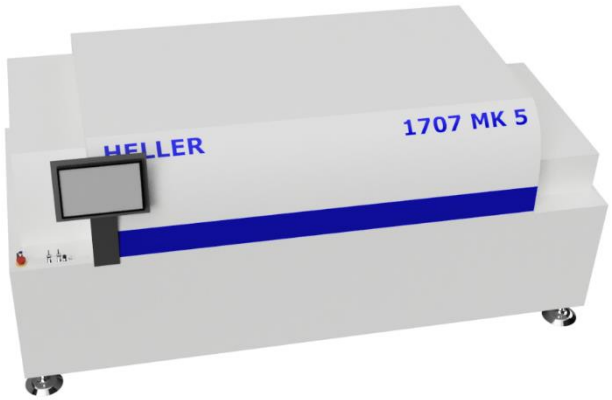
1 portalno postolje sa 6 glava	
Brzina postavljanja	30 000 CPH
Točnost ponavljanja:	$\pm$40 µm $Cpk \geq$ 1,0 (0402), 1 µm $Cpk \geq$ 1,0 (IC, Fiksna kamera)
Komponente na pomičnoj kameri:	0603~22 mm

Komponente na fiksnoj kameri:	55×55 mm, $L75$ mm, $H15$ mm
Maksimalna veličina panela:	460×400 (standardno), 740×460 (opcijonalno)

Programiranje ovog sustava se može raditi *offline* i *online* kako je objašnjeno u 1.

3.1.3. Peć za lemljenje

Odabrana peć za *reflow* lemljenje je HELLER 1707 MK 5 sustav (Slika 34). Značajnija karakteristika ove peći je 7 zona za grijanje.



Slika 34. HELLER 1707 MK 5, render 3D modela

Tablica 3. Karakteristike HELLER 1707 MK 5 [26]

Dimenzije:	$340 \times 137 \times 160$ cm
Zone grijanja:	7 gore i 7 dolje
Dužina grijanja:	182 cm
Zone hlađenja:	1 gore, dolje (opcionalno)
Potrošnja energije:	7 – 14 kW
Maksimalna brzina transportera:	188 cm/min

Kao i kod EKRA-e programiranje se radi *offline*.

3.1.4. Transporteri

Kako bi linija funkcionirala kao jedna cjelina potrebno je povezati sve elemente, što se radi različitim vrstama transportera. U ovom slučaju su odabrani: ASYS basic Loarder BUL

01, VEGO Compact BCO 01, VEGO Compact BCO 02 te ASYS basic Unloader BUL 01. Svaki od njih ima posebnu ulogu u SMT procesu.

1. ASYS basic Loader BUL 01

ASYS basic Loader BUL 01 (Slika 35) je transporter koji služi za utovar PCB panela u EKRA sustav. Utovarivač (engl. *loader*) ima mogućnost utovarivanja 50 panela nakon čega se držač PCB panela treba ponovno popuniti ili zamijeniti. *Loader* se može programirati za različiti raspored panela u držaču, npr. ukoliko se radi dvostrani PCB mora se staviti veći korak između pojedinog panela jer će u suprotnom u drugom prolazu zbog visine komponenti doći do kolizije. U slučaju mijenjanja koraka kapacitet držača se smanjuje.

Slika 35. ASYS basic Loader BUL 01, render 3D modela

2. VEGO Compact BCO 01

VEGO Compact BCO 01 (Slika 36) je kompaktni transporter za spajanje elementa i prenošenje PCB panela preko male udaljenosti.

Slika 36. VEGO Compact BCO 01, render 3D modela

3. VEGO Compact BCO 02

VEGO Compact BCO 02 (Slika 37) je transporter s dva različita dostavna lanca i više senzora za različite točke zaustavljanja.

Slika 37. VEGO Compact BCO 02, render 3D modela

Dva dostavna lanca služe za adaptaciju brzina, one dostavne prethodnog elementa te elementa kojemu se treba dostaviti. Peć ima znatno manju brzinu kojom se panel kreće kroz nju, a brzina ulaženja panela i ona kojom se panel kreće unutar peći moraju biti što sličnije. Isto vrijedi i na izlazu iz Hanwhe, koja ima znatno veću brzinu izbacivanja panela nego peć.

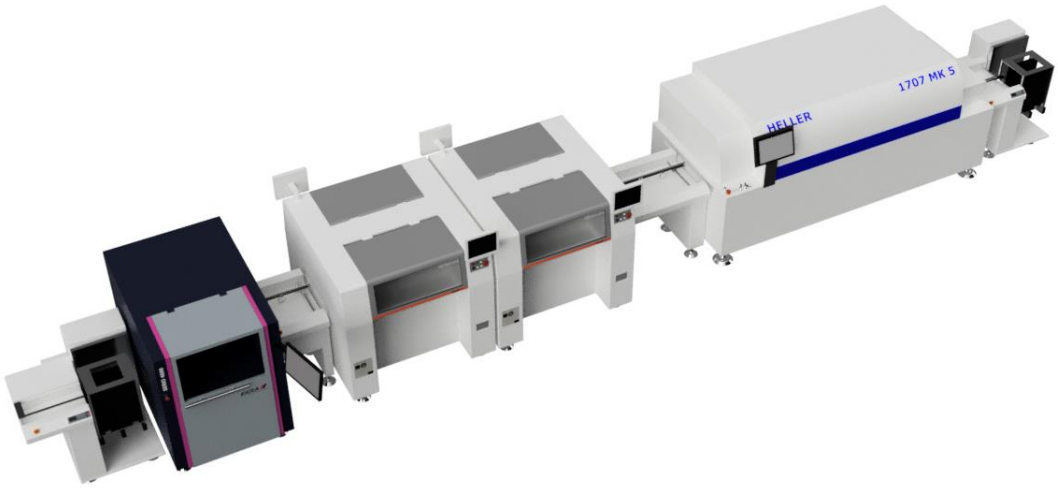
Kombinacija više senzora može poslužiti za brze inspekcije panela nakon izlaza iz *pick and place*-a, prije ulaska u peć.

4. ASYS basic Unloader BUL 01

Isto kao što je *loader* služio za utovarivanje panela, ASYS basic Unloader BUL 01(Slika 38) služi za istovar panela te slaganje istih u drugi držač panela.

Slika 38. ASYS basic Unloader BUL 01, render 3D modela

Svi navedeni elementi čine SMT postrojenje prikazano renderom na Slika 39.



Slika 39. Osmišljena SMT linija, render 3D modela

Proces rada započinje postavljanjem PCB panela u držače na *loader*-u (Slika 35). Pokretanjem Ekre (Slika 32), PCB paneli automatski ulaze u sustav gdje se panel automatski pozicionira vizijskim sustavom ili pinom, preko maske se nanosi lemna pasta te se vrši inspekcija. Ukoliko panel prolazi inspekciju Ekra ga prosljeđuje prvom transporteru (Slika 36), nakon čega ulazi u Hanwhu (Slika 33). Sklop *pick and place* sustava se sastoji od dva stroja kako bi se vrijeme postavljanja SMD-a umanjilo. Hanwha skenira *fiducial*-e[17] panela te počinje s postavljanjem komponenti. Završetkom rada oba stroja, panel se predaje drugom transporteru (Slika 37) koji vrši usporavanje panela kako bi izjednačio brzinu putanje s brzinom primanja Hellera (Slika 34). Unutar 7 toplinskih zona peći se razvija optimalan profil rastapanja te ponovnog zalemljivanja lemne paste. Završetkom lemljenja, panel se predaje *unloader*-u (Slika 38) koji ih posprema u držače. Izgled projektirane linije moguće je vidjeti na sljedećoj slici:

Slika 40. SMT proizvodna linija

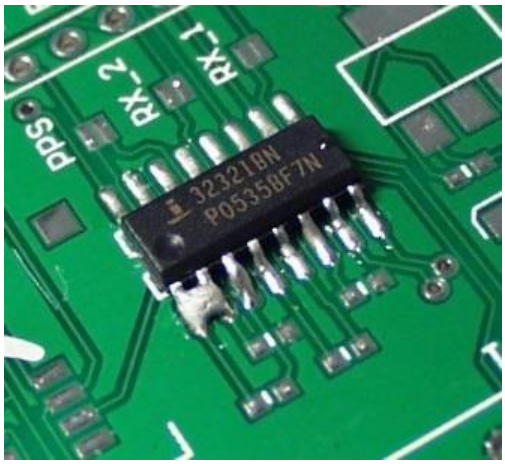
[17] *Fiducial* – referentne točke, s obzirom na koje se pozicionira sve ostalo u sustavu.

3.2. Problemi i greške

Prilikom rada projektirane linije, usprkos svim senzorima i kalibraciji SMT procesa, i dalje može doći do različitih grešaka i zastajanja zbog kojih krajnji PCB neće raditi. Greške mogu nastati prilikom svakog koraka u SMT proizvodnji, ali najčešće su one koje se događaju prilikom postavljanja SMD komponenti.

3.2.1. Premošćivanje lemom

Premošćivanje lemom (engl. *solder bridging*) je lemljenje preko dva ili više vodiča koji nisu električno povezani, čime se uzrokuje kratki spoj koji može rezultirati kvarom elektroničke pločice.



Slika 41. Premošćivanje lemom [27]

U SMT procesu, ovaj tip greške se najčešće događa prilikom printanja lemnom pastom, kada može doći do:

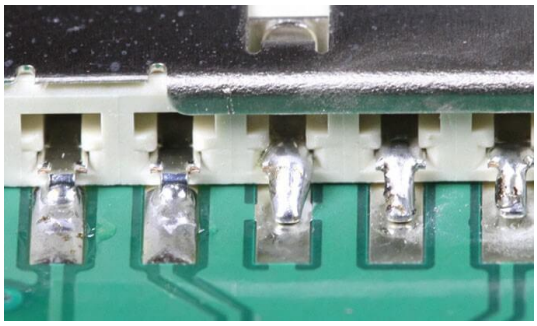
- pomaka ili zamaka maske u odnosu na panel, pri čemu se lemna pasta otisne na kriva mjesta, tj. između dva poluvodiča,
- postavljanja previše lemne paste, što se događa kada je maska loše postavljena u odnosu na panel po z osi, tj. jedno na drugo ne nasjeda dobro.

Osim problema prilikom printanja lemne paste, ova greška se može javiti i zbog:

- loše lemne paste (pasta s hladnim točkama ili pasta s lošim omjerom fluksa i lemnog metala),
- reflow procesa (ako je loš toplinski profil, odnosno previsoka temperatura, može doći do izlijevanja lema),
- loše točnosti postavljanja SMD komponenti.

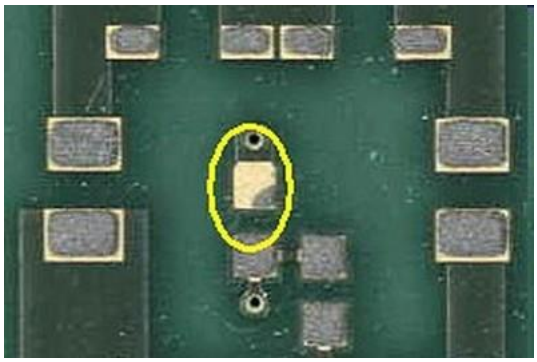
3.2.2. Manjak lemnog spoja

Manjak lemnog spoja se javlja kada zbog nedostatka lema vodič na PCB-u i nožica SMD-a nisu električno spojeni, tj. njima ne prolazi struja jer nisu u kontaktu.



Slika 42. Fizička odvojenost nožica i vodiča (treća, četvrta i peta nožica) [28]

Ovaj tip greške se također najčešće javlja prilikom procesa printanja lemne paste, u slučaju kada su otvori maske zatvoreni sasušenom pastom.



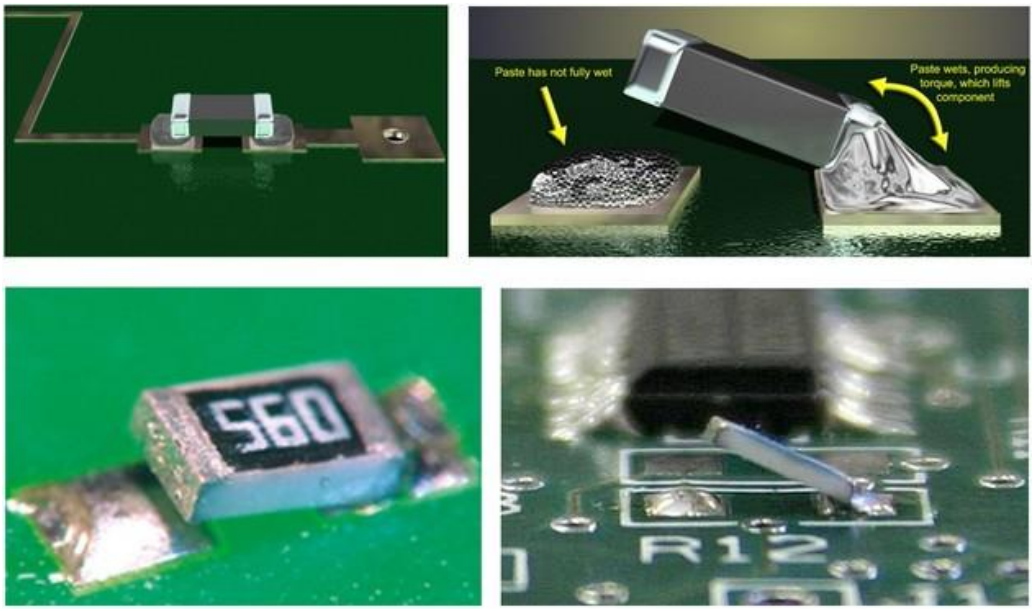
Slika 43. Manjak lemne paste

Dodatno, do slučaja da nožica i vodič nisu u kontaktu dolazi i zbog greške SMD komponente, odnosno ako neka od nožica ima defekt.

3.2.3. Tombstoning

Tombstone je SMD komponenta koja se djelomično ili potpuno vertikalno digla s vodiča PCB-a, pri čemu je ostala zalemljena na drugom vodiču (Slika 44).

Tombstoning se događa zbog nejednolične raspodjele topline prilikom *reflow* procesa, kada se lemna pasta rastapa u drugačijim vremenskim intervalima i povlači komponentu k sebi. Usto, ovaj tip greške javlja se ako je SMD krivo postavljen, tj. više prema jednom od vodiča.



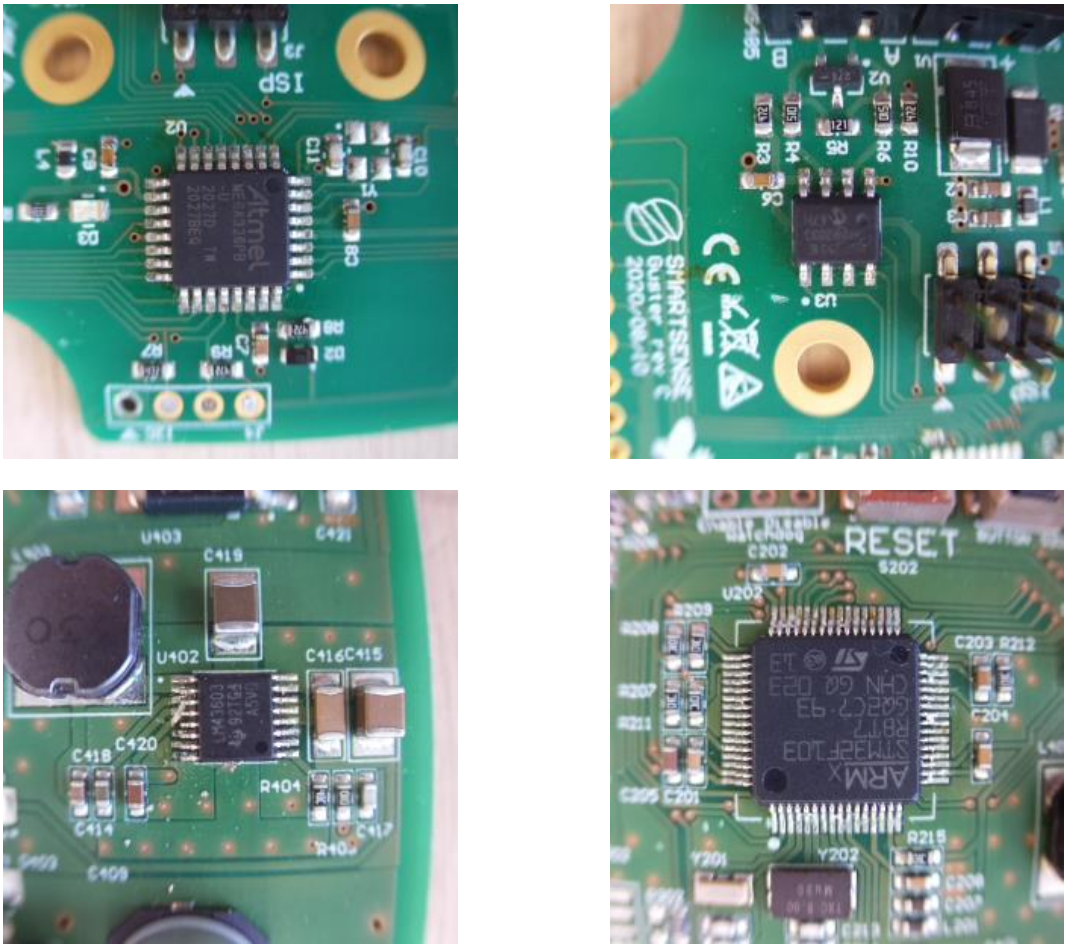
Slika 44. *Tombstoning*: dobro zalemljena komponenta (lijevo-gore), tombstoning (ostale)

3.2.4. Krivo orijentirane i pozicionirane komponente

Iskustveno, kriva orijentacija i pozicija neke komponente u x-y ravnini je najčešća greška koja se javlja u praksi, a može se dogoditi radi nekog od sljedeća tri razloga.

1. Tijekom programiranja komponenti određena je kriva orijentacija na izlazu iz hranilice. Ovo je ljudska pogreška, koja se može dogoditi zbog nepažnje, nepoznavanja komponenti, loše dokumentacije, i slično.
2. Zbog velike brzine rada stroja i male debljine PCB panela može doći do odskakanja SMD komponenti s panela.
3. Unutar hranilice se može nalaziti par komponenti drugačije orijentacije od ostalih, što je greška proizvođača koja se dogodila prilikom pakiranja komponenti.

Budući da kod većine otpornika i kondenzatora orijentacija nije bitna, ova je greška specifična za neke vrste čipova. Njima je na različiti način naznačena pravilna orijentacija: točka, manja točka (ukoliko su dvije), crta, skošenje, itd. Ako se prilikom inspekcije, koja prethodi pečenju, ne uoči greška, komponente će se krivo zalemiti i PCB neće raditi. Iako postoje brojne tehnike ručnog odlemljivanja i ponovnog zalemljivanja komponenti, često je slučaj da je to nemoguće izvesti (čipovi, male komponente, velike serije, itd.)



Slika 45. Greška krive orijentacije komponenti

3.3. Rješenje

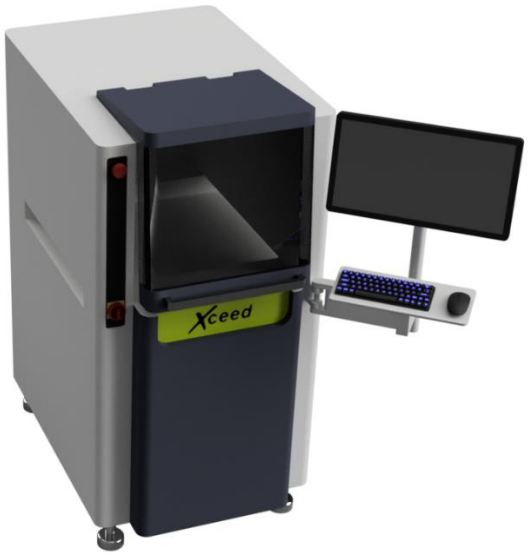
Bilo koju od spomenutih greški nije teško eliminirati, već ih je teško uočiti. Uočavanje se može odraditi povećanom pažnjom radnika na strojevima, no to često nije najpouzdanije rješenje pošto ljudska radna snaga nije u mogućnosti držati potpunu koncentraciju duži vremenski period. Uz to, treba uzeti u obzir troškove treniranja osoblja koje će moći prepoznati greške. Dok se neke greške može uočiti povećanom pažnjom i treningom radnika, ostale je izuzetno teško uočiti, čak i uz korištenje adekvatne opreme za manualnu optičku inspekciju koja uključuje: stereo mikroskope, mjerne sustave, digitalne sustave i različita povećala.

Ipak, danas se neke od grešaka mogu riješiti unutar sustava u kojem nastaju. Problemi nastali printanjem lemne paste se rješavaju dobrim postavljanjem sustava te vizijskim sustavom samoga stroja, čime on može uočiti ako dođe do stvrdnjavanja paste u otvorima maske, razmazivanja paste na panelu ili lošeg pozicioniranja maske s obzirom na PCB panel.

Problemi nastali prilikom *reflow* procesa se rješavaju različitom senzorikom koja prati temperaturni profil unutar peći. Ako je temperaturni profil u početku dobro namješten, šansa da dođe do greške je minimalna.

Kao što je već spomenuto, najčešće su greške koje se javljaju prilikom postavljanja komponenti i upravo je njih teško riješiti. Danas postoje *pick and place* sustavi koji imaju vlastite vizijske sustave i sami donekle mogu odraditi inspekciju, no takva inspekcija znatno usporava već najkompleksniji dio procesa. Iz tog razloga, česta je praksa postavljanja AOI sustava poslije *pick and place*-a, osim ako industrija ne zahtjeva drugačije[18].

Unutar ovog diplomskog rada, za unaprjeđenje osmišljene SMT proizvodne linije (Slika 39), odabran je PARMI Xceed AOI sustav (Slika 46) navedenih karakteristika (Tablica 4), također iz ponude tvrtke Amtest iz već spomenutih razloga.



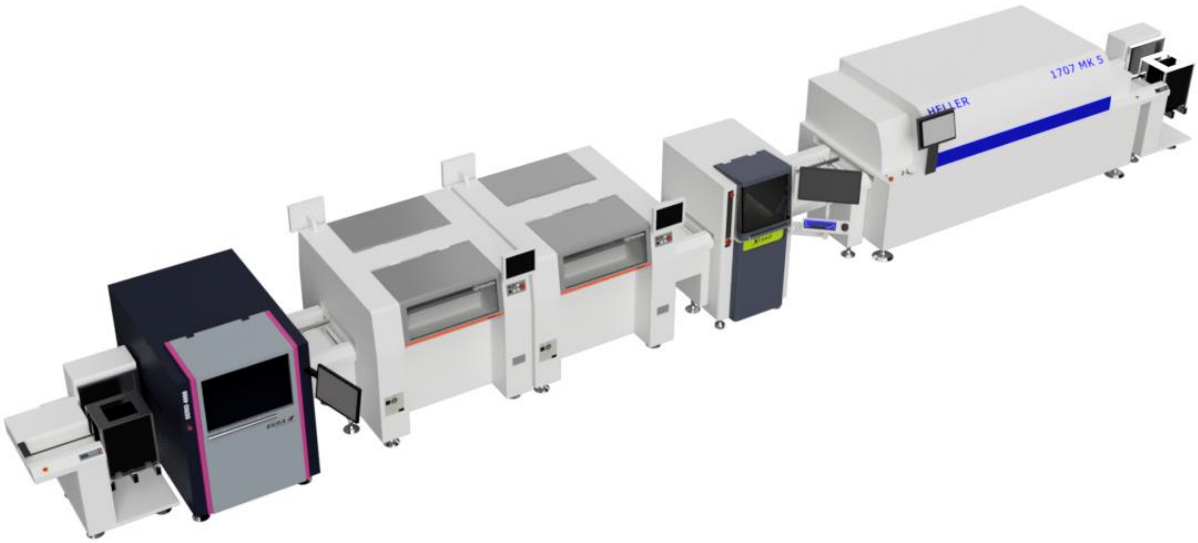
Slika 46. PARMI Xceed, render 3D modela

Tablica 4. Karakteristike PARMI Exceed sustava

3D AOI platforma sa visokom brzinom rada
Opremljena s inovativnim 3D senzorom teksturiranim u boji
Smanjeno vrijeme učenja pomoću pametnog algoritma
Intuitivno korisničko sučelje
Izuzetno niska stopa lažnih odziva i stopa bijega istinskim 3D oblikom

[18] Često je industrijski zahtjev za velike serije da se AOI sustav postavi poslije peći, kako bi napravio razvrstavanje loših od dobrih PCB panela.

Implementacijom ovog sustava u liniju dobiva se novi izgled postrojenja (Slika 47).



Slika 47. Osmišljena SMT linija sa implementiranim AOI sustavom

Na Slika 47 može se vidjeti i dodatak VEGO Compact BCO 01 transportera (Slika 36) koji je potreban zbog dodatka AOI sustava. Njegov dodatak produžio je ukupnu dužinu osmišljene proizvodne linije za otprilike 1500 mm. Pitanje je ima li dovoljno prostora za implementaciju tog rješenja. Također, pozicije instalacija su u ovom slučaju promijenjene, što također zahtjeva različite adaptacije.

Kako bi se izbjegli veliki i vremenski zahtjevni troškovi adaptacije, u sljedećem poglavlju opisan je razvoj i implementacija algoritma koji će moći zamijeniti AOI sustav, pri čemu neće biti potrebna adaptacija prostora.

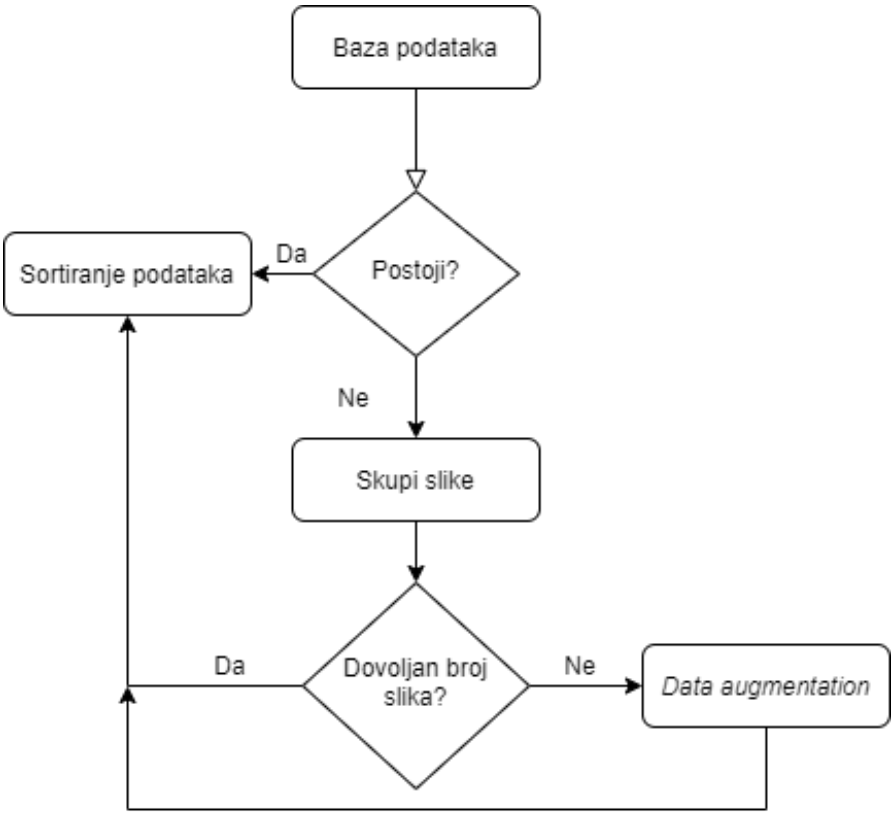
4. RAZVOJ I IMPLEMENTACIJA AOI ALGORITMA

Kao što je već spomenuto u poglavlju 2.2, postoje velike prepreke za razvoj algoritma konvencionalnim metodama. Te prepreke se mogu riješiti, ali rješenja su najčešće kompleksna i skupa. Iz tog razloga u ovom diplomskom radu takva su se rješenja izbjegla razvojem algoritma baziranog na konvolucijskim neuronskim mrežama. Algoritam je razvijen u programskom jeziku Python, korištenjem sljedećih modula: OpenCV, Tensorflow, Keras, Picamera.

- Python je objektno orijentiran, fleksibilan programski jezik opće namjene. Iako je znatno sporiji u odnosu na C i C++, ovdje će se koristiti zbog lakoće implementacije velikog broja paketa specijaliziranih za različite primjene.
- OpenCV (engl. *Open Source Computer Vision Library*) je knjižnica funkcija programiranja namijenjenih za programiranje računalnog vida. Napravljen je kako bi se ubrzalo korištenje računalnog vida u komercijalne svrhe. Unutar rada koristit će se zbog svoje jednostavnosti i brojnih mogućnosti koje će omogućiti manipulaciju slikama.
- Tensorflow je besplatna knjižnica, kao i OpenCV, koja služi za strojno učenje. Poseban joj je fokus na treniranju i zaključivanju neuronskih mreža.
- Keras djeluje kao sučelje za Tensorflow knjižnicu. Nastao je kako bi se smanjila kompleksnost programiranja neuronskih mreža čime se omogućila lakša i šira upotreba.
- Picamera je besplatna knjižnica koja pruža upravljanje Pi kamerom unutar Python programa.

Konvolucijske neuronske mreže također imaju svoje prepreke koje je potrebno razriješiti, a prva i najbitnija je sastavljanje kvalitetne i opširne baze podataka.

4.1. Pripremanje baze podataka



Slika 48. Dijagram toka pripremanja baze podataka

Za skupljanje slika potreban je adekvatan sustav za prikupljanje slika ili se preuzima već postojeća baza podataka. Budući da baze podataka namijenjene za ove svrhe nisu javno dostupne potrebno je izraditi vlastitu. Kako bi slike u bazi podataka bile što konzistentnije za ovaj diplomski rad razvijen je sustav koji će uzimati slike. Sustav se sastoji od Raspberry Pi kamere, Raspberry Pi 3 računala, Stepcraft-2/D.420 hobističkog CNC-a te adekvatnih adaptera koji sve spajaju u jednu cjelinu.

- Raspberry Pi High Quality Camera odabrana je zbog jednostavnosti korištenja i dostatne preciznosti. Specifikacije kamere se mogu vidjeti u [30].

Slika 49. Raspberry Pi High Quality Camera [30]

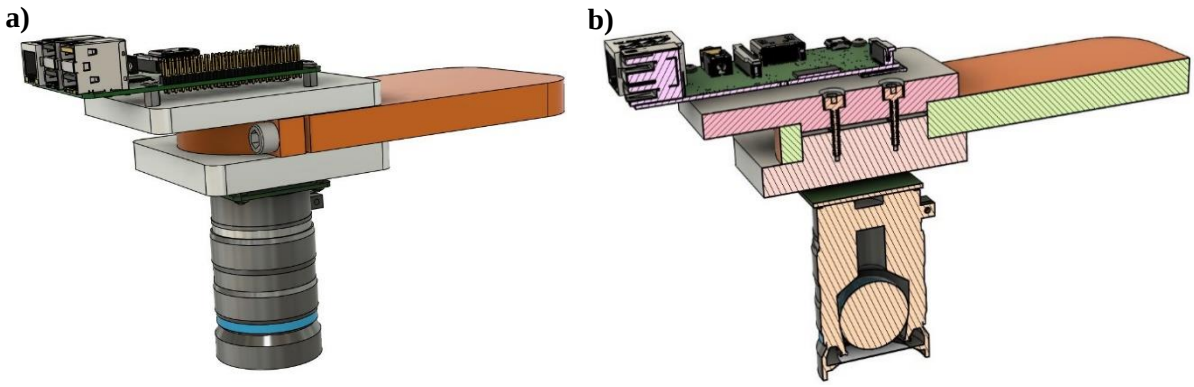
- Raspberry Pi 3 (RP3) je serija malih jedno-pločnih računala, od kojih se jedno koristi u svrhu upravljanja kamerom i procesiranjem podataka. Specifikacije modela Pi 3 Model B korištenog za upravljanje moguće je vidjeti na [30].

Slika 50. Raspberry Pi 3

- Stepcraft-2/D.420 je hobistički CNC stroj na čiji nosač se može staviti niz različitih alata: ruter, laser, nož, olovka, itd. U sklopu ovog diplomskog rada, koristi se radi lakšeg i preciznijeg pozicioniranja kamere, koja se preko adaptera pričvrsti na nosač stroja. Karakteristike stroja je moguće vidjeti na [32].

Slika 51. Stepcraft-2/D.420 [32]

- Adapter je razvijen u dva dijela, prema tehničkoj dokumentaciji u prilogu I, a s obzirom na dimenzije kamere i dostupnog materijala. 3D model sklopa prikazan je na Slika 52, a stvarni izgled adaptera je vidljiv na Slika 53.



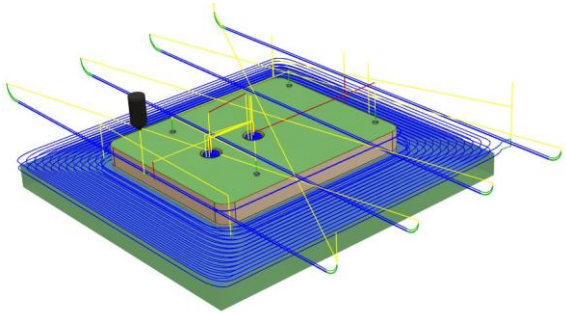
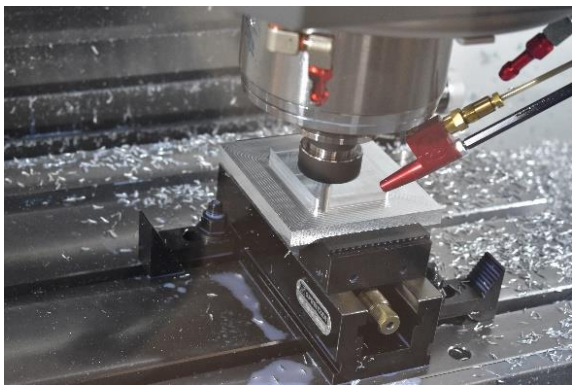
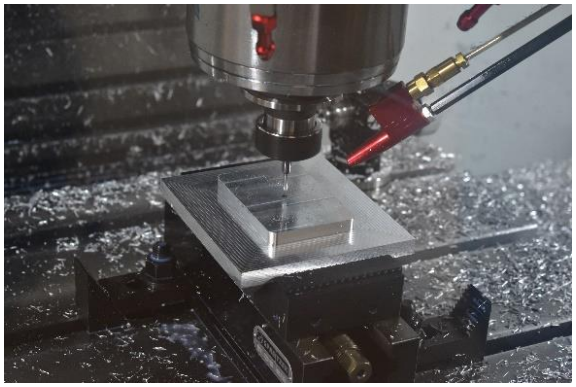
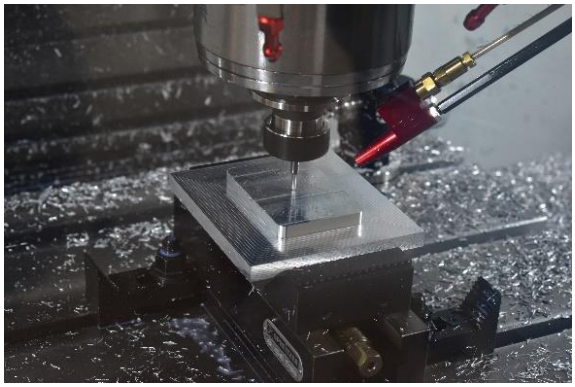
Slika 52. 3D model: a) adaptera u sklopu i b) u presjeku

Slika 53. Stvarni sklop

Adapter je izrađen od aluminija EN AW 2017 na HAAS VF-2 CNC stroju. Putanje alata za obradu napravljene su u Fusion 360 software-u, nakon čega se putanje pretvaraju u G-kod[19] za CNC stroj. U Tablica 5 prikazane su slike nastale prilikom izrade adaptera.

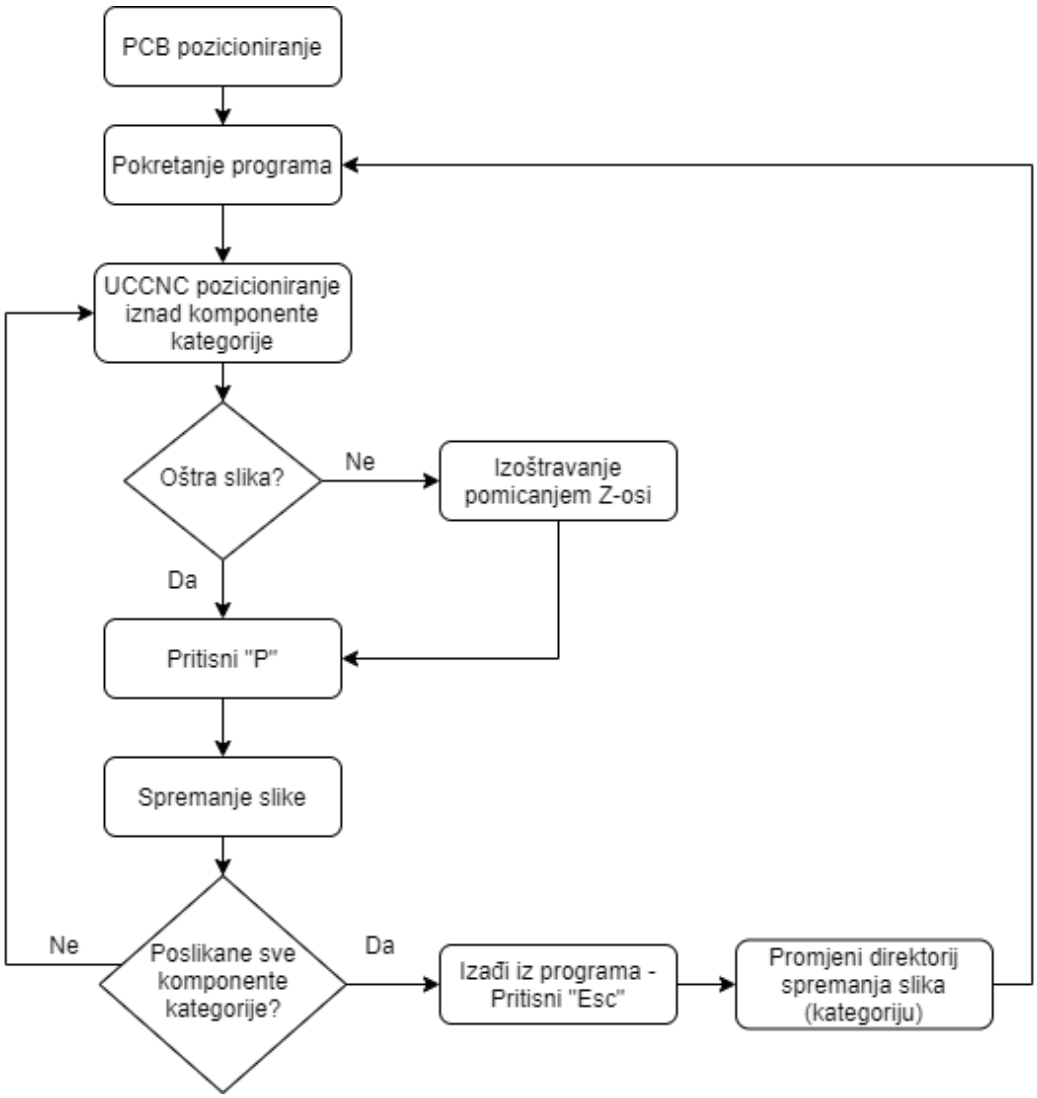
[19] G-kod su naredbe računalnog programa za CNC upravljanje.

Tablica 5. Simulacija putanje alata (a), izrada konture (b), bušenje (c i d)

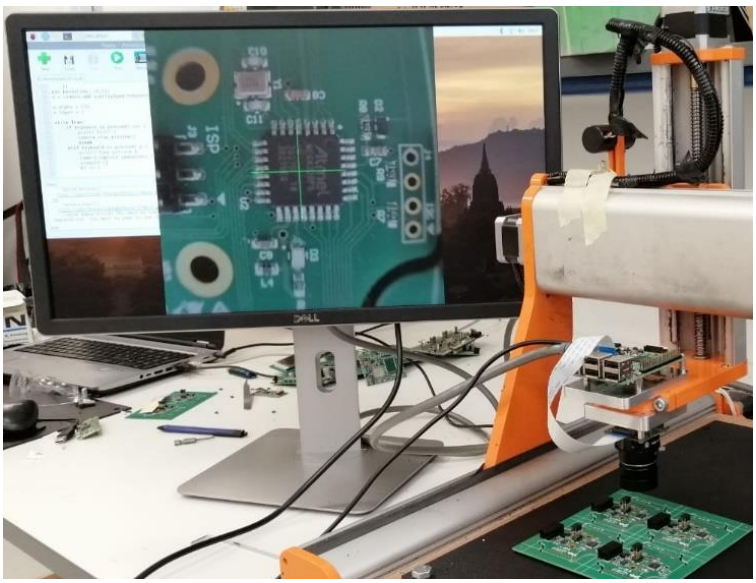
 **a)**	 **b)**
 **c)**	 **d)**

Sustav radi (Slika 54) tako da se PCB pozicionira na radnu površinu CNC stroja, nakon čega se software-om UCCNC20 postiže oštrina slike (pomicanjem z osi) te dolazi na određenu poziciju od interesa (pomicanjem x/y osi). Potom se programom na RP3 uzimaju slike i spremaju u određenu kategoriju (Slika 55).

[20] UCCNC – vrsta programa za upravljanje strojevima.



Slika 54. Dijagram toka rada sustava skupljanja podataka za bazu podataka



Slika 55. Proces uzimanja slike

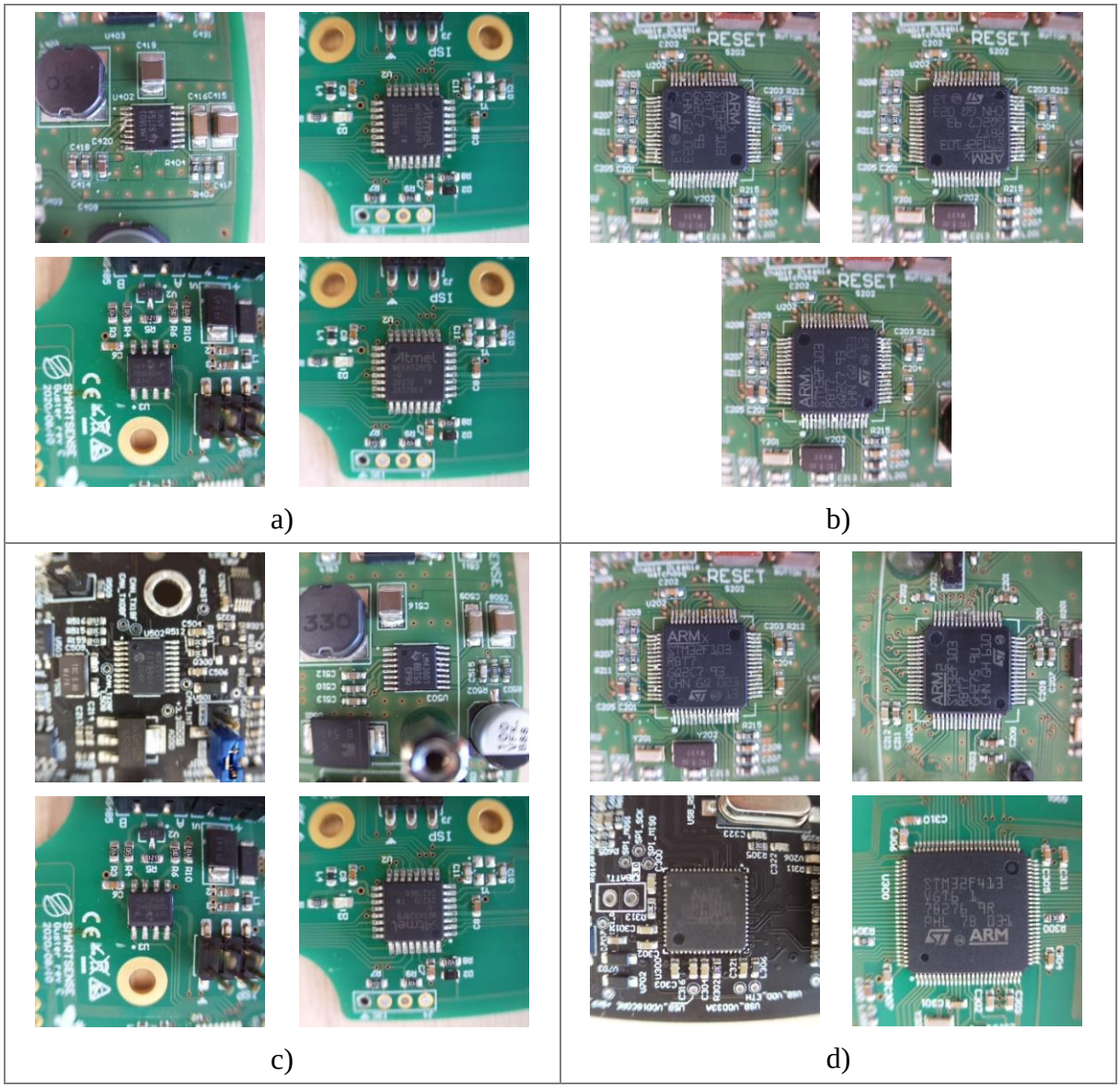
Program za navedeno, napisan u Python-u, vidi se na Slika 56. Inicijalizacija kamere i rezolucije snimanja se radi preko linija koda 6 do 9. Program za uzimanje i spremanje pojedine slike u bazu podataka nalazi se u linijama 22 do 31. Tu je inicijalizirano da se slika uzima pritiskom tipke „P“ na tipkovnici, koja se potom sekvencijalno sprema u datoteku svaki put pod novim imenom. Pritiskom tipke „Esc“ izlazi se iz programa. Prilikom uzimanja slika pojedinih kategorija bitno je promijeniti direktorij spremanja slika kako se isto ne bi trebalo raditi kasnije.

```
1  from picamera import PiCamera
2  from time import sleep
3  from PIL import Image
4  import keyboard
5
6  br = 1
7  camera = PiCamera()
8  camera.resolution = (300, 300)
9  camera.start_preview()
10 img = Image.open('Slike/kriz2.png')
11
12 pad = Image.new('RGB', (
13     ((img.size[0] + 31) // 32) * 32,
14     ((img.size[1] + 15) // 16) * 16,
15 ))
16 pad.paste(img, (0, 0))
17 o = camera.add_overlay(pad.tobytes(), size=img.size)
18
19 o.alpha = 130
20 o.layer = 3
21
22 while True:
23     if keyboard.is_pressed('esc'):
24         print('Exit!')
25         camera.stop_preview()
26         break
27     elif keyboard.is_pressed('p'):
28         print('Take pitcure')
29         camera.capture_sequence(['/home/pi/Desktop/DIPLOMSKI/Slike/01/%02d.jpg' % br])
30         sleep(0.5)
31         br += 1
```

Slika 56. Python kod za uzimanje slika

U svrhu ovog diplomskog rada određene su četiri kategorije od interesa: dobro orijentirane komponente s jednom točkom (kategorija: „GOOD01“), loše orijentirane komponente s jednom točkom (kategorija: „BAD01“), dobro orijentirane komponente s dvije točke (kategorija: „GOOD02“) i loše orijentirane komponente s dvije točke (kategorija: „BAD02“). Neke od slika iz kategorije je moguće vidjeti u Tablica 6.

Tablica 6. Primjeri iz baze podataka: a) BAD01, b) BAD02 , c) GOOD01 i d) GOOD02



Unutar baze podataka skupljeno je 6 slika kategorije BAD01, 3 slike kategorije BAD02, 20 slika kategorije GOOD01 i 4 slike kategorije GOOD02. Primjećuje se nejednolik raspored slika unutar baze podataka, zbog čega mreža može naučiti zaključivati u korist onih podataka kojih ima više. No tu se javlja i jedan mnogo veći problem, baza podataka sadrži premalo podataka, zbog čega može doći do *overfitting*-a[21]. Da bi se mreža uspješno istrenirala potrebno je barem desetak tisuća slika po kategoriji. Kako bi se zaobišla oba problema koristile su se sljedeće dvije tehnike: *data augmentation* i *transfer learning*. Za jednostavniji trening mreže, podaci su se sortirali i razdijelili na podatke za trening, validaciju i testiranje.

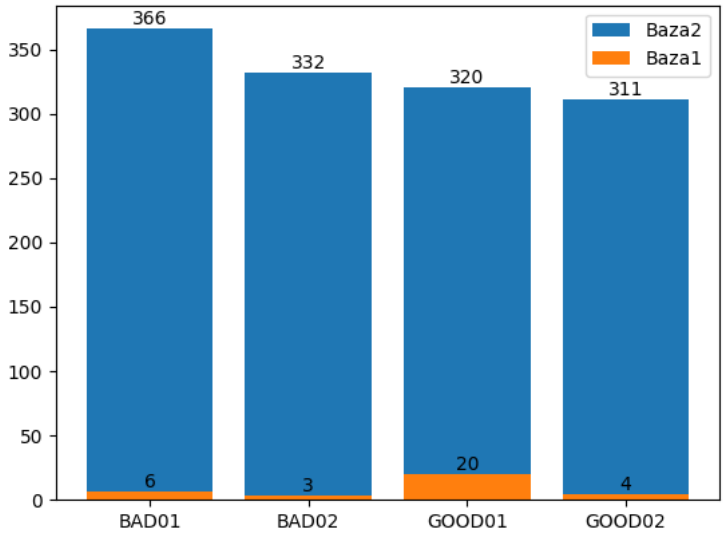
[21] *Overfitting* – objašnjen u poglavlju 4.2.

- *Data augmentation* je tehnika povećavanja baze podataka u procesima analize podataka, dodavanjem neznatno izmijenjenih kopija ili novostvorenih sintetičkih podataka iz postojećih podataka. Korištena je u svrhu povećanja baze podataka te manipulacije podataka unutar svake kategorije kako bi oni bili u korist učenja algoritma. Programirano je da se za loše okrenute komponente uvijek generira više slika kako bi algoritam u slučaju nesigurnosti pogodio da se radi o greški. Iako će ovo smanjiti točnost algoritma u smislu pogađanja dobrih komponenti, povećati će točnost pogađanja grešaka.

```
gen = ImageDataGenerator(rotation_range=10,
                         width_shift_range=0.1,
                         height_shift_range=0.1,
                         shear_range=0.15,
                         zoom_range=0.1,
                         channel_shift_range=10.,
                         horizontal_flip=True
                         )

PATH = 'Slike/Data5'
os.chdir(PATH)

for i in os.listdir():
    br = 1
    size = int(300 / len(os.listdir(f'{i}')))
    for j in os.listdir(f'{i}'):
        image = np.expand_dims(plt.imread(i + '/' + j), 0)

        if i == 'GOOD01' or i == 'GOOD02':
            for k in range(size):
                p = "AI-" + str(br) + "-"
                aug_iter = gen.flow(image,
                                    batch_size=10,
                                    save_to_dir=i,
                                    save_prefix=p,
                                    save_format='jpg',
                                    )
                aug_iter.next()
        else:
            for k in range(size+10):
                p = "AI-" + str(br) + "-"
                aug_iter = gen.flow(image,
                                    batch_size=10,
                                    save_to_dir=i,
                                    save_prefix=p,
                                    save_format='jpg',
                                    )
                aug_iter.next()
        br += 1
```

Slika 57. Python kod za augmentaciju baze podataka

Varijablom „gen“ sa Slika 57 su definirana ograničenja augmentacije, kao što su: maksimalna rotacija slike, zamućenje, itd. Linijama koda od 23 do 49 se vrši manipulacija i spremanje novonastalih slika kako bi dobili prethodno navedenu bazu podataka.



Slika 58. Graf raspodjele podataka u bazi podataka prije augmentacije (Baza1) i poslije (Baza2)

Augmentacija se ne smije prekomjerno koristiti jer dolazi do stvaranja baze podataka s velikom količinom sličnih slika, stoga se koristi i tehnika *Transfer learning*-a obrađena u poglavlju 4.2.

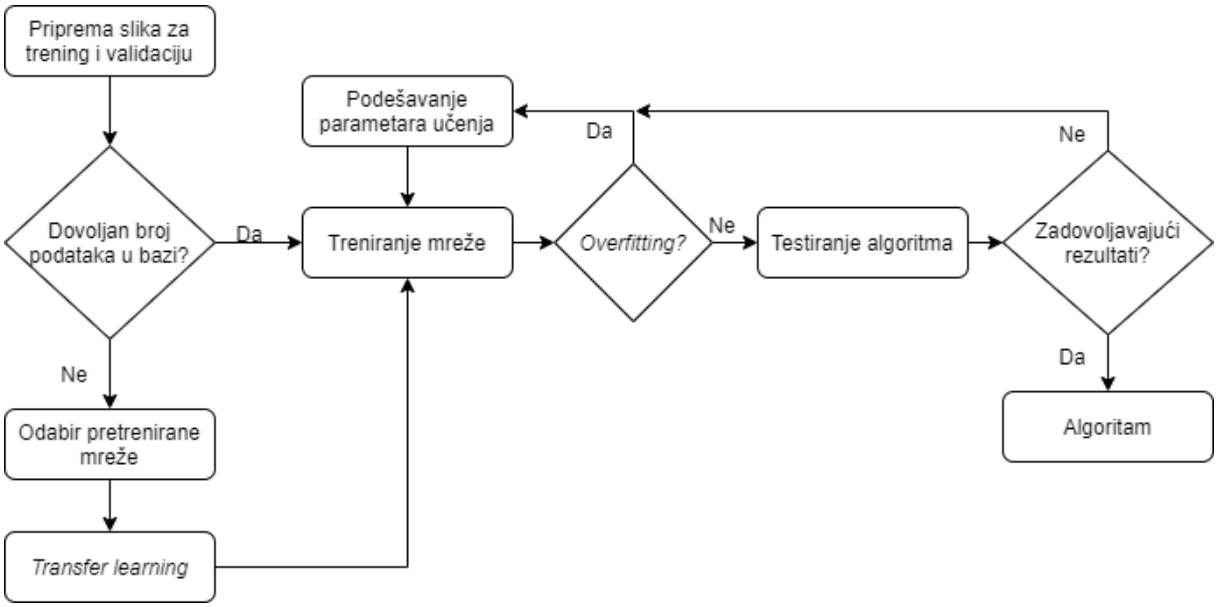
- Sortiranje podataka se odradilo pomoću paketa os i shutil, a prikazano je na Slika 59.

```
60      for i in CATEGORIES:
61          shutil.move(f'{i}', 'train')
62          os.mkdir(f'valid/{i}')
63          os.mkdir(f'test/{i}')
64
65          valid_samples = random.sample(os.listdir(f'train/{i}'), 40)
66          for j in valid_samples:
67              shutil.move(f'train/{i}/{j}', f'valid/{i}')
68
69          test_samples = random.sample(os.listdir(f'train/{i}'), 10)
70          for k in test_samples:
71              shutil.move(f'train/{i}/{k}', f'test/{i}')
72  os.chdir('../..')
```

Slika 59. Python kod za pripremu baze podataka

Slike se sortiraju na način da su unutar svake kategorije stvorene datoteke *train*, *valid* i *test*, pri čemu se sve slike iz date kategorije premještaju u *train* datoteku. Potom se nasumično odabire određen broj slika (ovisno o količini podataka) koje se premještaju u datoteke *valid* i *test*.

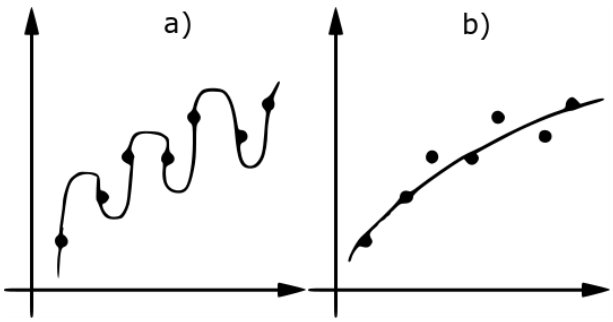
4.2. Razvoj algoritma



Slika 60. Dijagram toka razvoja algoritma

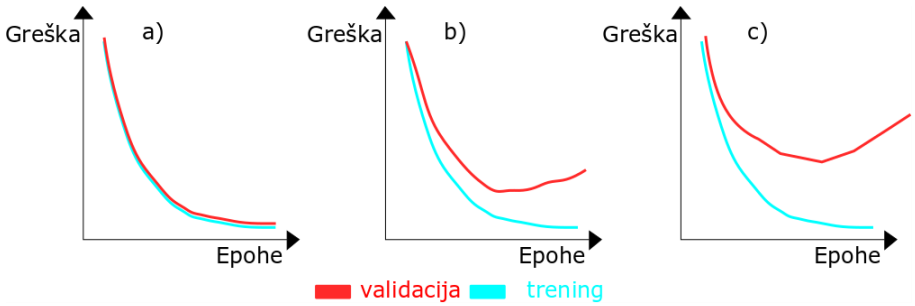
Kao što je prethodno spomenuto, kako bi se kompenzirao manjak slika u bazi podataka i spriječio *overfitting*, razvoj algoritma radio se *Transfer learning* metodom.

- *Overfitting* je koncept u statistici koji se javlja kada se statistički model uklapa točno u podatke iz treninga. Time model gubi mogućnost predviđanja novih podataka s kojima se još nije susreo.



Slika 61. a) *Overfitt* model i b) Optimalan model

U konvolucijskim neuronskim mrežama se *overfitting* definira kao učenje ulaznih slika napamet, pri čemu se gubi mogućnost raspoznavanja novih iz iste kategorije. *Overfitting* se u CNN analizira kao graf ovisnosti gubitaka treninga i validacije. Gubici treninga indiciraju koliko dobro se model poklapa s podacima za trening, a gubici validacije koliko se poklapa s podacima za validaciju (novim podacima). U pravilu bi gubici validacije trebali pratiti gubitke treninga i težiti prema nuli (Slika 62).



Slika 62. Prikaz *overfitting*-a: a) optimum, b) i c) *overfitting*

Optimum se postiže kada postoji dovoljno velik broj različitih podataka da mreža nauči karakteristike pojedine kategorije, a ne uči napamet slike iz te kategorije. Također treba obratiti pažnju i na broj epoha[22] koje se zadaju te ratu učenja mreže koji uz loše podatke mogu utjecati na *overfitting*.

- *Transfer learning* je istraživački problem gdje se već utrenirana mreža, s pohranjenim znanjem, koristi u svrhu učenja iz novih manjih baza podataka. Problem kojeg rješava postojeća mreža mora se moći povezati s problemom od interesa. Pritom se ne trebaju koristiti svi slojevi odabrane mreže već samo oni koji unutar slike pronalaze karakteristike od interesa, npr. konture, oblici itd.

```
# Odabir CNN-a
res = tf.keras.applications.resnet50.ResNet50()

# Stvaranje novog modela
x = res.layers[-1].output
output = Dense(units=4, activation='softmax')(x)
model = tf.keras.Model(inputs=res.input, outputs=output)

# Određivanje slojeva koji se mogu trenirati
for layer in model.layers[:-39]:
    layer.trainable = False

# Karakteristike treniranja
model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

Slika 63. *Transfer learning* python

[22] Epoha – označava jedan krug u učenju mreže.

Utrenirana mreža se odabire na način prikazan na Slika 63, red 38. Nakon odabira CNN-a, dodjeljuju se svi slojevi te mreže svojoj novoj mreži (linija 41, Slika 63), osim zadnjega koji služi za kategorizaciju podataka. Potom se mreži dodjeljuje sloj za kategorizaciju podataka (linija 43). Linijama 47 i 48 se određuje do koje dubine se želi omogućiti slojevima učenje težina. U ostatku koda se određuju karakteristike učenja mreže.

Cijeli Python kod za razvoj algoritma moguće je vidjeti u prilogu I pod nazivom „Algoritam“. Kod sadrži i potrebnu pripremu podataka te spremanje algoritma pod posebnim imenom kako bi se naknadno mogao koristiti u AOI svrhe.

4.3. Analiza rezultata

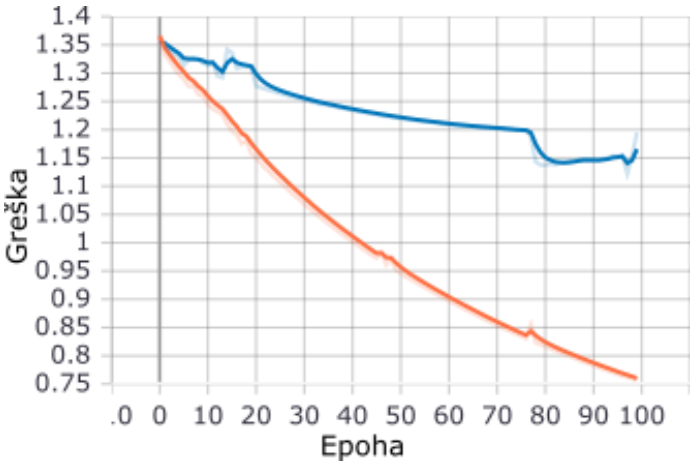
Kako bi se saznalo kako odabir pojedinih parametara utječe na učinkovitost pojedine mreže, potrebno je analizirati *overfitting* te napraviti konfuzijsku matricu[23] (engl. *confusion matrix*) za testne primjere. Kod pomoću kojega su se testirali algoritmi može se vidjeti u prilogu I pod nazivom „Algoritam_testiranje“. Sve analize su rađene nad istom bazom testnih podataka za 100 epoha, pri čemu je analizirano pet različitih algoritama. Parametri koji su se pritom mijenjali su: baza podataka, veličina serije (engl. *batch size*) slika za trening i validacija, dubina sloja učenja[24] te stopa učenja. Cilj analize je odabrati najoptimalniji algoritam koji će u najvećem broju slučajeva pogoditi koja je komponenta loše okrenuta, čak ako pritom dobro orijentiranu komponentu svrsta u kategoriju loših.

- Algoritam 1 napravljen je pomoću početne baze< podataka koja je sadržavala svega 31 sliku. Parametri primijenjeni u algoritmu se mogu vidjeti u Tablica 7.

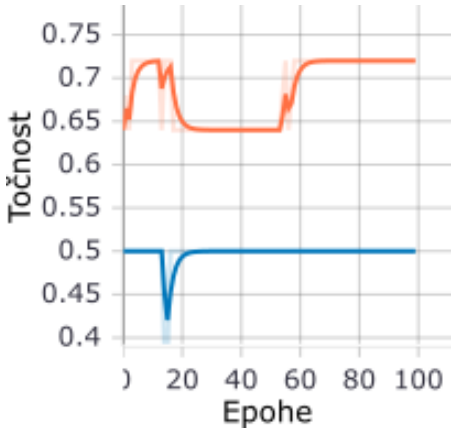
Tablica 7. Parametri algoritma 1

Baza podataka	Baza1 (Slika 58)
Veličina serije slika za trening	1
Veličina serije slika za validaciju	1
Dubina sloja učenja	-23
Stopa učenja	0,0001

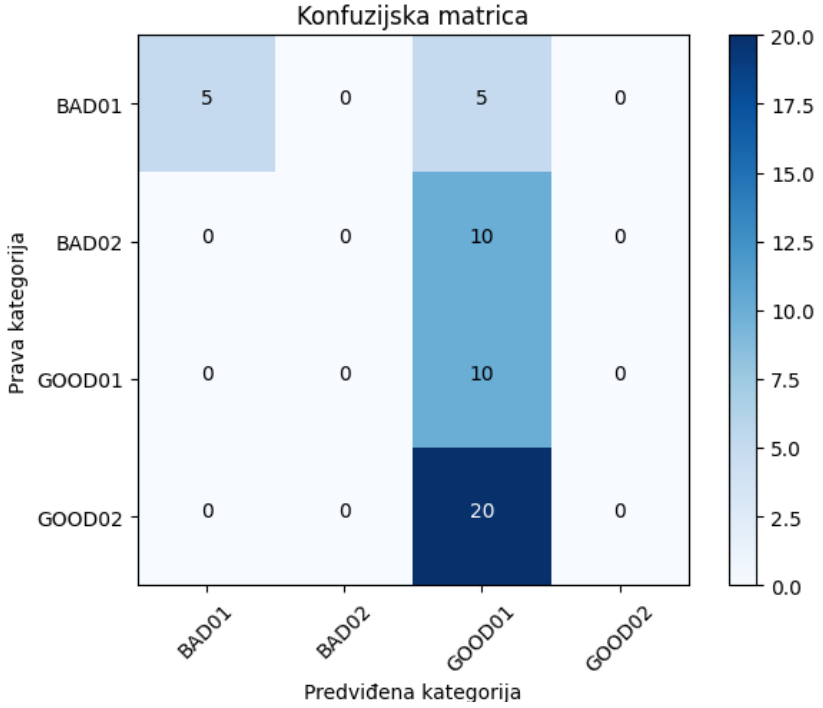
[23] Konfuzijska matrica je matrica koja omogućava vizualizaciju učinkovitosti algoritma.
[24] Dubina sloja učenja se definira na način da se gleda od zadnjeg sloja CNN arhitekture prema prvome, pri čemu se određuju slojevi čije će težine biti ponovno učene, dok će ostale ostati zamrznute.



Slika 64. Algoritam 1: greška treninga (narančasto) i greška validacije (plavo)



Slika 65. Algoritam 1: točnost treninga (narančasto) i točnost validacije (plavo)



Slika 66. Algoritam 1: Konfuzijska matrica

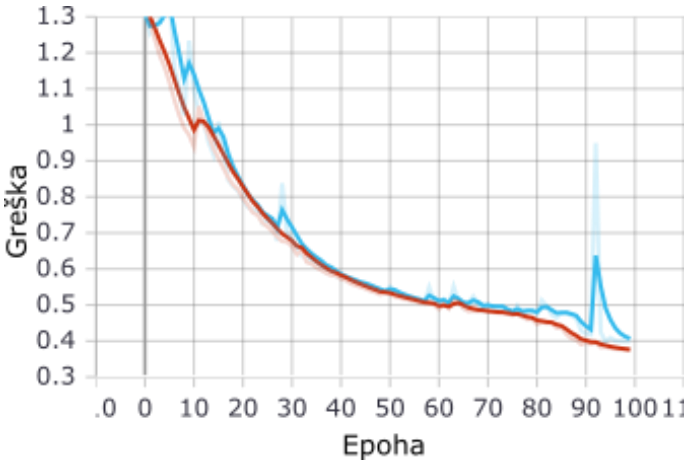
Na Slika 64 može se vidjeti pojavljivanje *overfitting*-a. Pad greške validacije ne prati padanje greške treninga što ukazuje na to da baza podataka ima premalo podataka. Ako bi se algoritam pustio da uči više od zadanih 100 epoha, moglo bi doći do učenja trening podataka napamet što rezultira padom točnosti validacije i rastom točnosti treninga. Navedeno se još ne događa (Slika 65). Slika 66 prikazuje da algoritam pogađa kategoriju testnih podataka s točnošću od 30% (točno pogođenih 15 od 50 podataka), gdje je točnost za pogađanje loše orijentirane komponente još manja, 25%.

Sve navedeno ovaj algoritam čini neupotrebljivim u AOI svrhu.

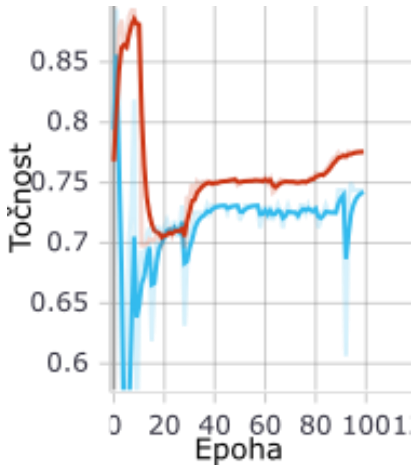
- Algoritam 2:

Tablica 8. Parametri algoritma 2

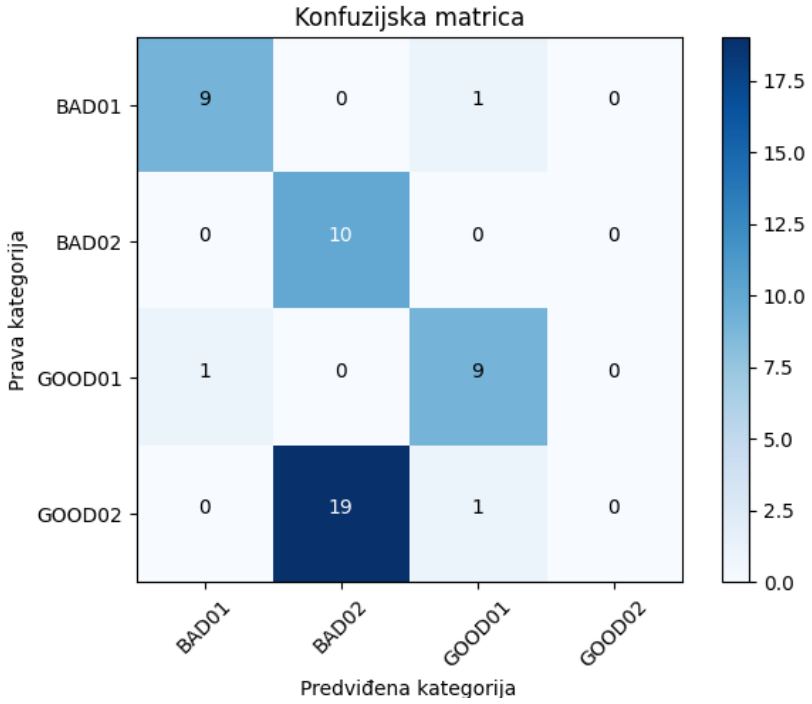
Baza podataka	Baza2 (Slika 58)
Veličina serije slika za trening	30
Veličina serije slika za validaciju	8
Dubina sloja učenja	-23
Stopa učenja	0.001



Slika 67. Algoritam 2: greška treninga (crveno) i greška validacije (plavo)



Slika 68. Algoritam 2: točnost treninga (crveno) i točnost validacije (plavo)



Slika 69. Algoritam 2: Konfuzijska matrica

Ako se usporede Slika 64 i Slika 67, vidi se koliko zapravo povećanje baze podataka pridonosi smanjenju javljanja *overfitting*-a. No, kako se vidi na Slika 68, previsoka stopa učenja, uz prevelike serije za trening i validaciju, dovodi do prenagle promjene težina što za rezultat ima izbacivanje pretreniranog algoritma iz globalnog minimuma. U slučaju adekvatne baze podataka to bi bilo dobro, jer bi za specifične nove podatke mreža tražila novi globalni minimum, no u ovom slučaju to nije tako. Na Slika 69 moguće je uočiti kako algoritam pogađa s točnošću od 56% što je bolje nego u prethodnom slučaju. Također, superiornost ove baze podataka nad prošlom moguće je vidjeti u podatku da je algoritam,

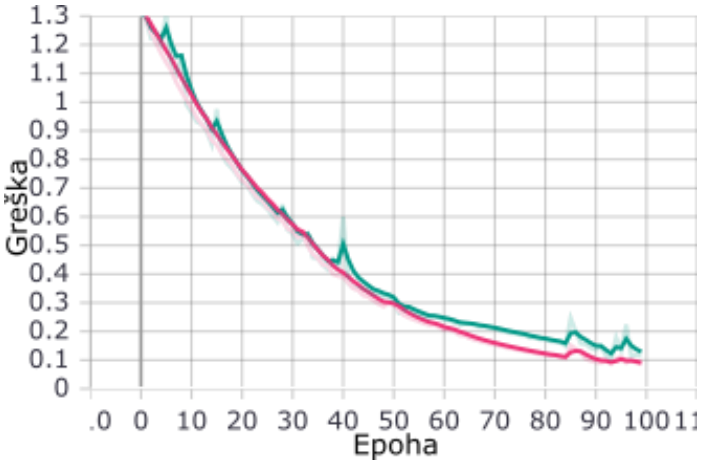
od 20 loše orijentiranih komponenata, samo jednu krivo pogodio čime ima uspješnost pogađanja loših komponenti od 95%.

S obzirom da uz 95% točnosti za pogađanjem, postoji 19 dobrih komponenata koje bi se morale i ručno provjeriti, ovaj algoritam i dalje ne zadovoljava zahtjeve za AOI.

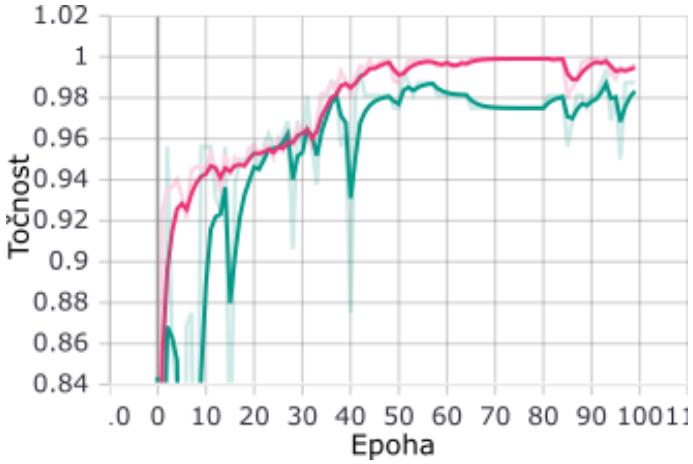
- Algoritam 3:

Tablica 9. Parametri algoritma 3

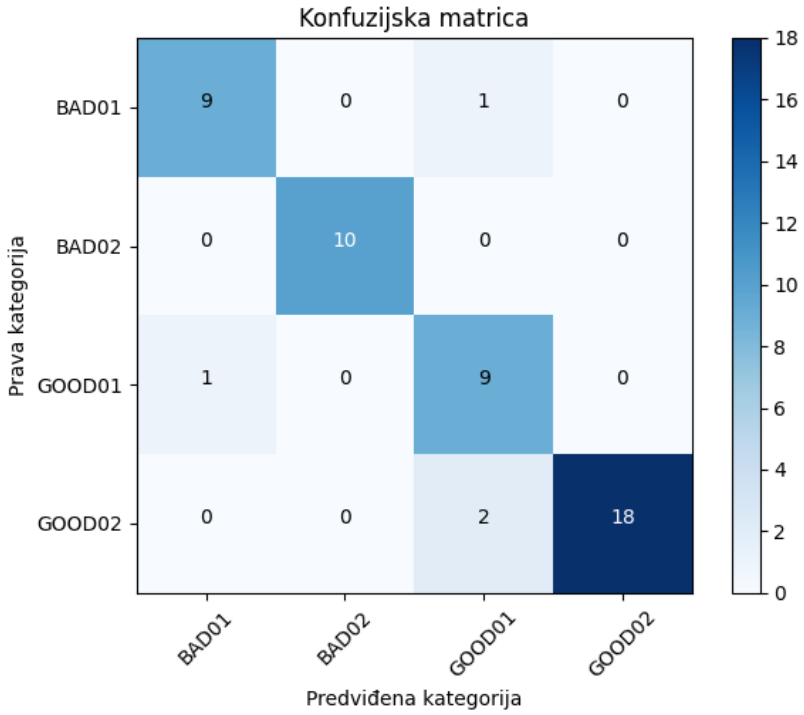
Baza podataka	Baza2 (Slika 58)
Veličina serije slika za trening	20
Veličina serije slika za validaciju	4
Dubina sloja učenja	-23
Stopa učenja	0.0005



Slika 70. Algoritam 3: greška treninga (rozo) i greška validacije (zeleno)



Slika 71. Algoritam 3: točnost treninga (rozo) i točnost validacije (zeleno)



Slika 72. Algoritam 3: Konfuzijska matrica

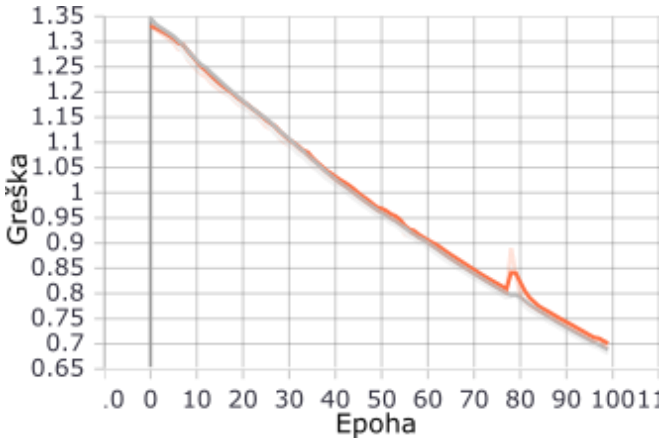
Na Slika 70 i Slika 71 moguće je uočiti poboljšanje u odnosu na prethodno dobiveni algoritam. Smanjenje veličine serija za treniranje i validaciju te smanjenje stope učenja je smanjilo mogućnost izlaženja iz globalnog minimuma. Iz konfuzijske tablice algoritma (Slika 72) se vidi kako algoritam ima točnost pogađanja 92% što je znatno više nego u prošlim slučajima.

Iako mu je točnost pogađanja dobrih komponenti 100%, zbog točnosti pogađanja loših komponenti od 95% niti ovaj algoritam ne odgovara implementaciji u AOI.

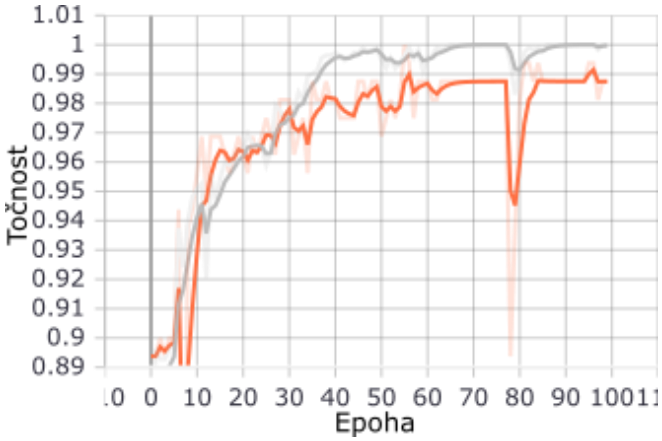
- Algoritam 4:

Tablica 10. Parametri algoritma 4

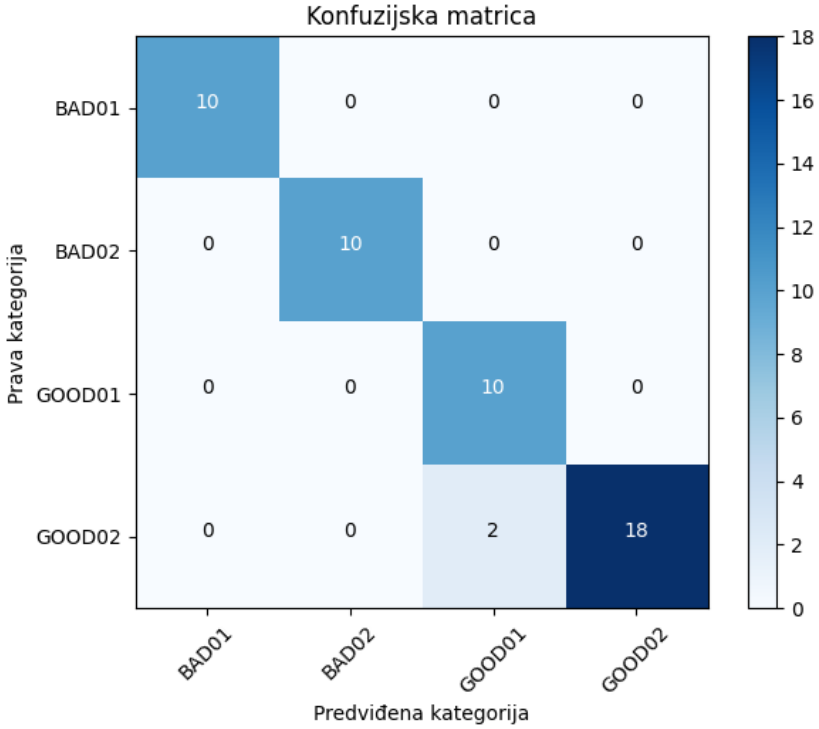
Baza podataka	Baza2 (Slika 58)
Veličina serije slika za trening	20
Veličina serije slika za validaciju	4
Dubina sloja učenja	-23
Stopa učenja	0.0001



Slika 73. Algoritam 4: greška treninga (sivo) i greška validacije (narančasto)



Slika 74. Algoritam 4: točnost treninga (sivo) i točnost validacije (narančasto)



Slika 75. Algoritam 4: Konfuzijska matrica

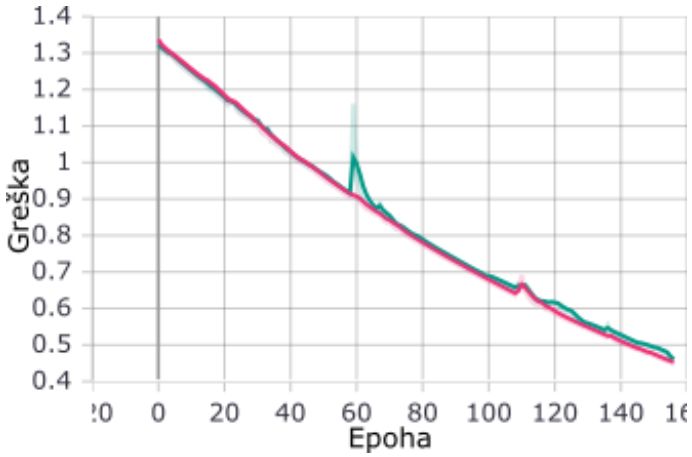
Daljnjim smanjenjem stope učenja dolazi do boljeg poklapanja greške validacije s greškom treninga (Slika 73), no unutar 100 epoha to rezultira njihovim većim iznosom nego u prošlim primjerima. Uspoređujući Slika 74 i Slika 71 vidi se kako je i točnost validacije i treninga nešto viša od onih trenutnog algoritma, no promatranjem konfuzijske matrice (Slika 75) se vidi povećanje točnosti pogađanja podataka za trening na 96%. Uz to sada je točnost pogađanja loših komponenti 100%, isto kao i dobrih[25], čime se za potrebe ovog zadatka može reći kako ovaj algoritam zapravo ima točnost pogađanja 100%. Iz navedenog se vidi kako se uz sve dostupne alate, konvolucijske neuronske mreže i dalje ne mogu u potpunosti razumjeti. Iz prikazanih grafova grešaka i točnosti se vidi kako bi se ovdje trebalo raditi o lošije postavljenom algoritmu u odnosu na prošli slučaj. Međutim, to ne mijenja činjenicu da se ovaj algoritam poklapa s ciljem postavljenim na početku poglavlja te se iz tog razloga odabire za AOI.

- Algoritam 5 je razvijen u svrhu postizanja validacijske i trening točnosti od 100% prilikom treninga. To se radi ograničavanjem treninga da prilikom postizanja navedenih točnosti prestane s treningom bez obzira na zadani broj epoha.

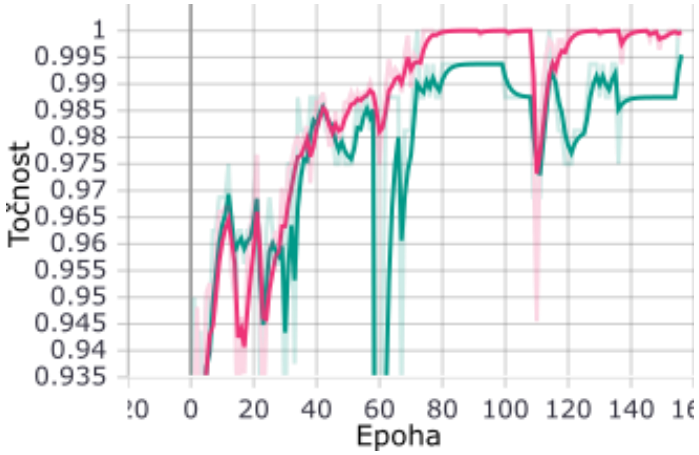
Tablica 11. Parametri algoritma 5

Baza podataka	Baza2 (Slika 58)
Veličina serije slika za trening	20
Veličina serije slika za validaciju	4
Dubina sloja učenja	-39
Stopa učenja	0.0001
Epohe	∞ - zaustavi kada točnosti budu 100%

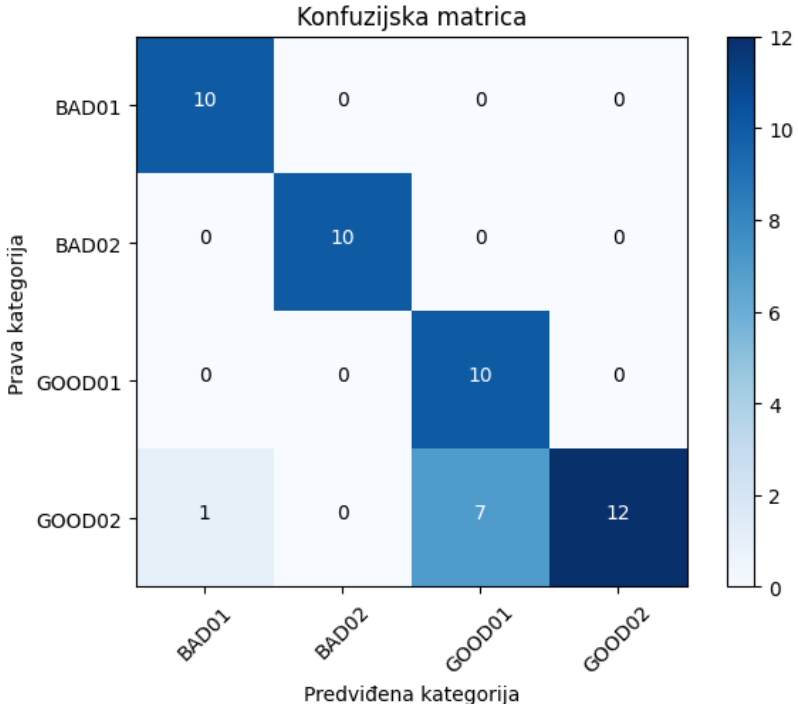
[25] Za razvoj sustava nije bitno nalazi li se dobra komponenta u kategoriji GOOD1 ili GOOD2, već samo da se naznači da je ista dobra.



Slika 76. Algoritam 5: greška treninga (rozo) i greška validacije (zeleno)



Slika 77. Algoritam 5: točnost treninga (rozo) i točnost validacije (zeleno)

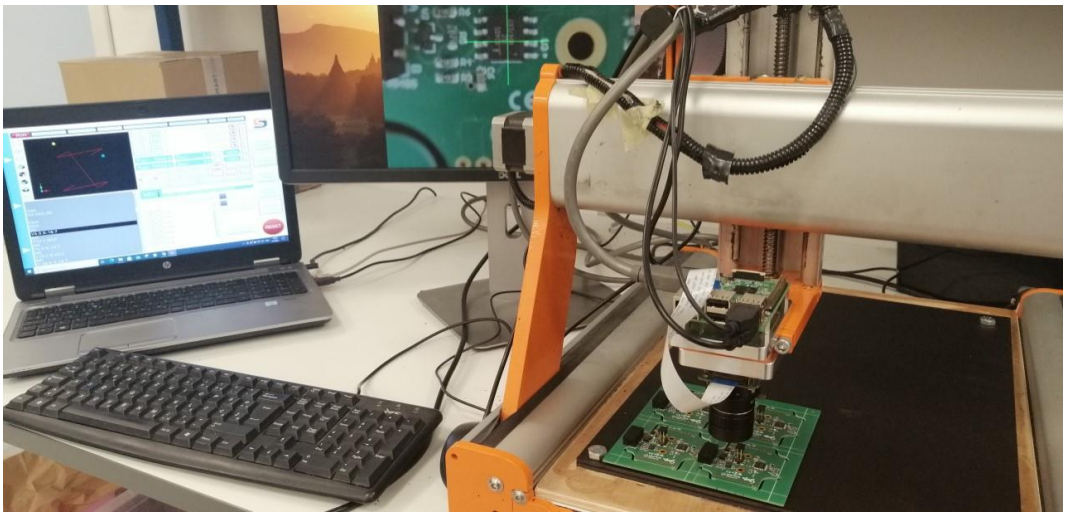


Slika 78. Algoritam 5: Konfuzijska matrica

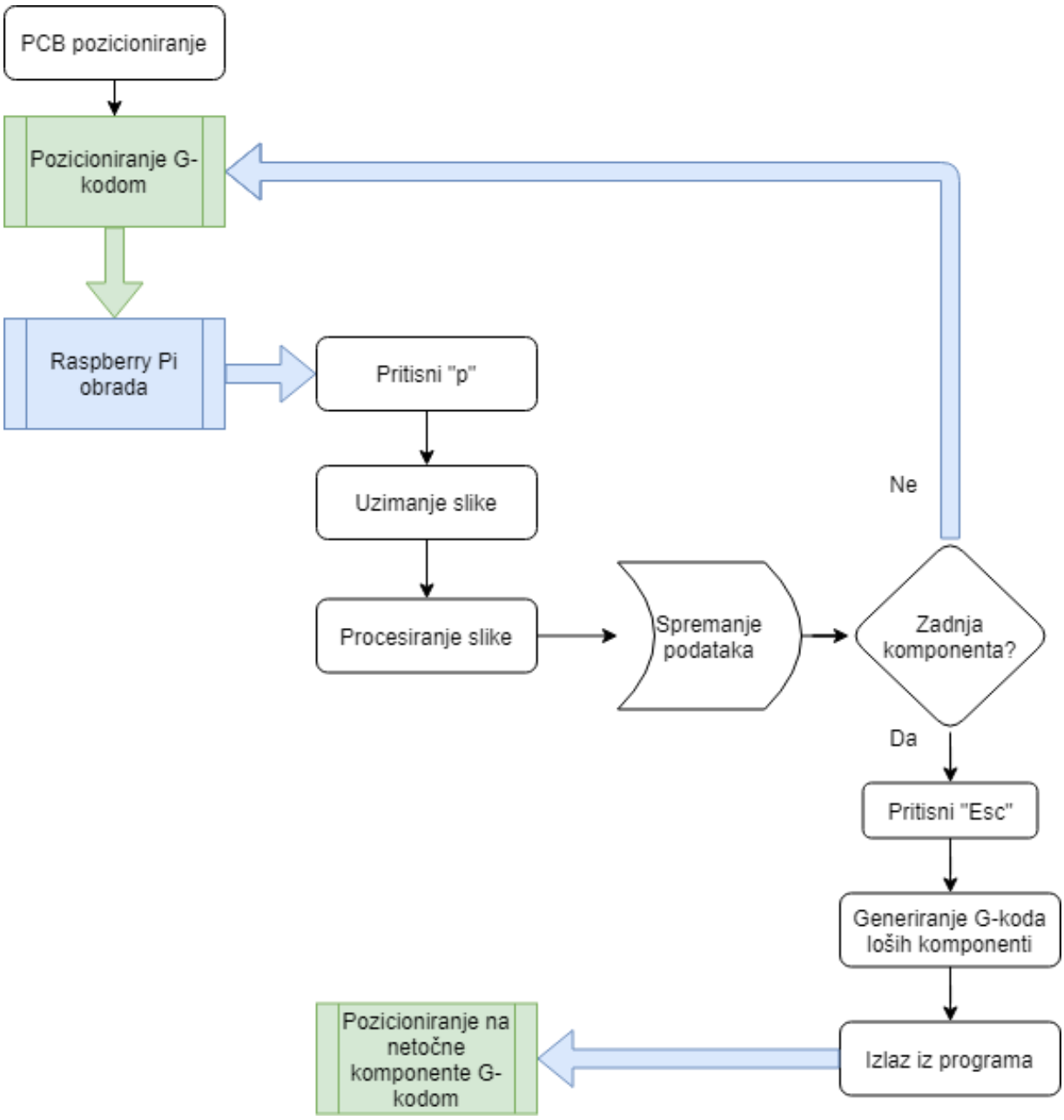
Iz Slika 77 moguće je vidjeti kako je treniranjem algoritam uspio doći do 100% točnosti unutar 140 epoha, čime se, u usporedbi na ostale algoritme, može reći kako je ovaj najbolji. Ipak, promatranjem konfuzijske matrice primjećuje se kako to nije slučaj. Iako je točnost pogađanja loše orijentiranih komponenata ostala 100%, pogađanje po kategorijama se smanjilo na 84%. Može se pretpostaviti kako je algoritam zapravo pretreniran te je zbog nejednakosti kategorija u bazi podataka krenuo pogađati u korist kategorija s više podataka u trening bazi, što može biti i rezultat povećanja dubine sloja učenja.

4.4. Implementacija

Kako bi osmišljeni AOI sustav bio kompletan odrađena je implementacija odabranog algoritma u sustav koji je korišten za stvaranje baze podataka, no sada u svrhu testiranja komponenata na PCB panelu. Proces radi prema dijagramu toka prikazanom na Slika 80. Panel se pozicionira na radnu površinu CNC-a. Tada se pokreće UCCNC program unutar kojeg se određuje ishodište koordinatnog sustava panela. Potom se pokreće G-kod te se kamera pozicionira iznad prve komponente. Pritiskom tipke „p“ na tipkovnici uzima se slika i šalje na obradu. Procesirana slika se potom sprema zajedno s predviđenom kategorijom. Proces „pozicioniranje G-kodom“ i „Raspberry Pi obrada“ se naizmjenično rade za svaku komponentu dok se ne dođe do zadnje pozicije u G-kodu. Pritiskom tipke „Esc“ pokreće se analiza koja sprema poziciju svih loših komponenti u formi G-koda, nakon čega se izlazi iz programa. Zadnji korak u procesu je ubaciti novi G-kod u UCCNC software kako bi se kamerom pokazalo koje komponente su loše okrenute.



Slika 79. Rad AOI sustava



Slika 80. Dijagram toka osmišljenog AOI sustava

- Pozicioniranje G-kodom

Kako bi se kamera pozicionirala točno iznad komponente od interesa potrebno je znati njenu točnu koordinatu. Koordinate pojedinih komponenti, s obzirom na proizvoljno ishodište određeno u programu za dizajn PCB-a, dobiva se unutar *Pick and place.xlsx* datoteke namijenjene za *pick and place* stroj, tipa Hanwha. Unutar programa moguće je odrediti koje informacije će biti dostupne u izvezenoj datoteci, a za ovu svrhu su potrebne informacije o designiranom mjestu, x koordinata te y koordinata. Datoteku excel formata je

potom potrebno učitati i iz nje izvući sve bitne podatke, koji se potom pretvaraju u g-kod. Python kod koji radi tu pretvorbu je moguće vidjeti na Slika 81 i Slika 82.

```
1   import pandas as pd
2   import os, sys
3
4   lista = {}
5   lista2 = []
6   # dezignirana mjesta komponenti od interesa
7   komponente = ['U2', 'U3']
8   br = 0
9
10  # pozicioniranje u datoteku unutar koje će se raditi obrada
11  os.chdir('P&P_data')
12
13  # otvaranje excel datoteke
14  df = pd.read_excel('PickAndPlace_Guster 1.xlsx')
15
16  # kreiranje g-kod datoteke
17  fd = os.open('P&P_Guster.nc', os.O_RDWR|os.O_CREAT)
18
19  # kreiranje početnih komamndi g-koda
20  #G90-apsolutna distanca, G0-linearna interpolacija sa maksimalnom brzinom,
21  # G53-linearna interpolacija u koordinatnom sustavu stroja
22  # G64-kontrola putanje po konstantnoj brzini
23  # G54-work offset
24  os.write(fd, bytes('G90\nG0 G53 Z0.\n\nG64\nG54\nZ0.',
25                     encoding='utf8'))
26
27  # definiranje odstupanja za panel
28  array = [0, 0, -56, 0, 0, 62, -56, 62]
```

Slika 81. Pretvorba u G-kod 1

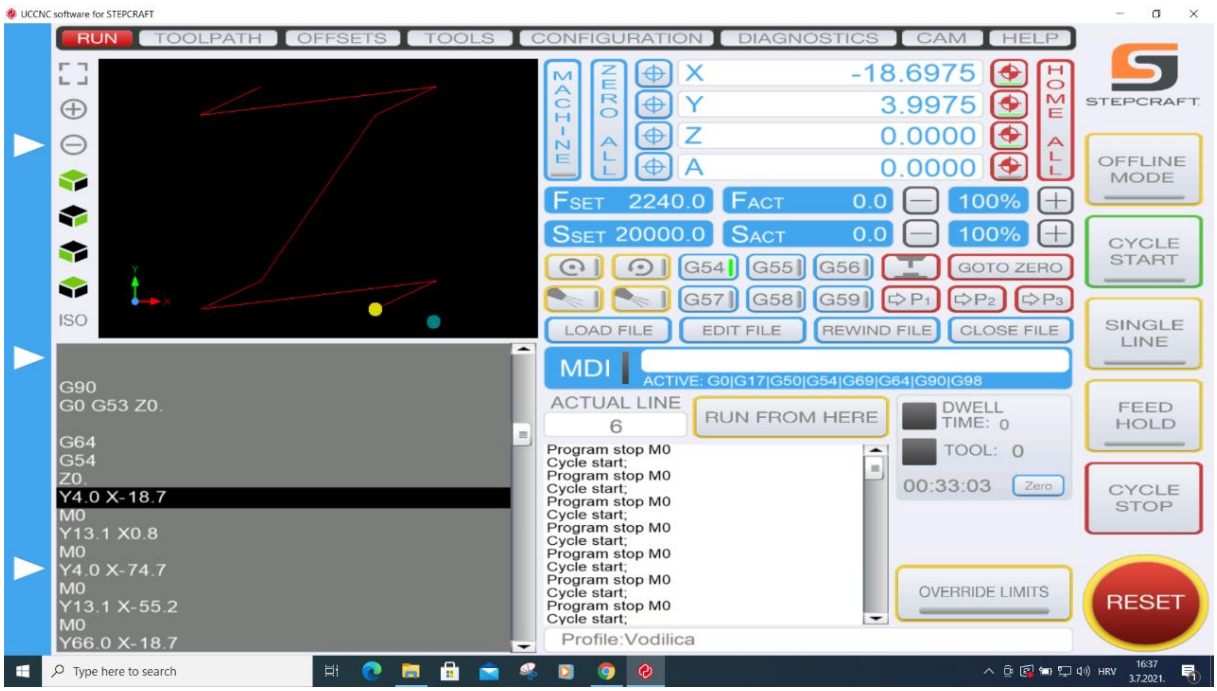
Slika 81 prikazuje prvi dio Python koda unutar kojeg se odabiru komponente od interesa (linija 7), nakon čega se radi pozicioniranje u datoteku unutar koje će se raditi sve obrade, tj. čitanje i pisanje datoteka (linija 10). Potom se *Pick and place* datoteka učita pod varijablom po izboru te se otvori nova datoteka pod nekom varijablom u koju će se vršiti pisanje G-koda (linije 14 i 17). Linije 20 do 23 opisuju osnovne G naredbe koje trebaju biti na početku G-koda, a linija 24 radi upisivanje tih G naredbi u datoteku. Također je potrebno definirati i razmake PCB-a unutar panela, s obzirom da *Pick and place* datoteka sadrži samo koordinate komponenata na PCB-u.

```
30  # izvlačenje koordinata iz excel datoteke i prebacivanje u g-kod
31  for j in range(int(len(array)/2)):
32      for i in range(len(df.values)):
33          if df.values[i][0] in komponente:
34              # vrijednost y koordinate
35              Y_val = str(float(df.values[i][1]*-1 + array[br*2+1]))
36              # vrijednost x koordinate
37              X_val = str(float(df.values[i][2] + array[br*2]))
38              # kreiranje g-koda
39              X = bytes(("\nY" + Y_val + " X" + X_val + "\nM0"),
40                        encoding='utf8')
41              os.write(fd, X)
42      br += 1
43  # završavanje g-koda
44  os.write(fd, bytes('\nM30',
45                             encoding='utf8'))
46
```

Slika 82. Pretvorba u G-kod 2

Slika 82 prikazuje obradu Excel datoteke i pisanje G-koda. Prolazi se za svaki PCB na panelu po cijeloj Excel datoteci pri čemu se čitaju samo komponente od interesa (linija 33). Tada se, ovisno o prethodno određenim odstupanjima PCB-a, unutar panela određuju koordinate na način da se na postojeće očitane podatke dodaju odstupanja.

Dobiveni G-kod se potom ubacuje u software UCCNC na računalo koje upravlja CNC strojem, čime se isti upravlja (Slika 83).



Slika 83. UCCNC software sa G-kodom u prozoru lijevo dolje

- Raspberry Pi obrada

Ovaj proces podrazumijeva učitavanje algoritma i svih potrebnih ulaznih podataka, uzimanje slike, procesiranje slike, spremanje informacija te generiranje G-koda komponenata s lošom orijentacijom kako je prikazano na Slika 80. Glavni dio koda koji obavlja zadatak obrade slike se vidi na Slika 84.

```
55 # Procesiranje
56 def obrada():
57     print('obrada start')
58 
59     # Preprocesiranje i pretvaranje slike u array
60     slika = tf.keras.preprocessing.image.load_img(path, target_size=(IM_SIZE, IM_SIZE))
61     array1 = tf.keras.preprocessing.image.img_to_array(slika)
62     array1 = np.array([array1])
63 
64     # Predviđanje rezultata
65     predictions = model.predict(array1)[0]
66     print(predictions, max(predictions))
67 
68     # Kategorizacija rezultata
69     for i in range(len(predictions)):
70         if predictions[i] == max(predictions):
71             if i in BAD:
72                 predikcija.append(KATEGORIJE[1])
73                 img = Image.open(path2 %br)
74                 ImageDraw.Draw(img).text((20, 20), KATEGORIJE[1], fill=(255, 0, 0), font=font)
75                 img.save(path2 %br)
76             elif i in GOOD:
77                 predikcija.append(KATEGORIJE[0])
78                 img = Image.open(path2 %br)
79                 ImageDraw.Draw(img).text((20, 20), KATEGORIJE[0], fill=(0, 255, 0), font=font)
80                 img.save(path2 %br)
81     print(predikcija)
82     return predikcija
```

Slika 84. Raspberry Pi obrada slike

Proces se odvija na način da se učita i preprocesira slika, prethodno uhvaćena kamerom, nakon čega se pretvara u matricu. Potom se radi predikcija odabranim algoritmom. Kako bi rezultati predviđanja bili što razumljiviji, linijama 69 do 81 se radi selekcija rezultata s obzirom da je rješenje predikcije postotak pripadanja pojedinoj kategoriji (Slika 85).

```
[0.17209637 0.16468318 0.50665456 0.15656595]
```

Slika 85 Predviđanje algoritma

Uz to se svaka obrađena slika uredi na način da se preko nje ispiše „GOOD“ ili „BAD“ ovisno o tome je li orijentacija dobra ili loša (Slika 86).

a) **b)**

Slika 86. Uređene slike poslije obrade: a) Dobro orijentirana komponenta, b) loše orijentirana komponenta

Dio koda koji naknadno radi generiranje G-koda isti je kao i na Slika 81 i Slika 82, uz dodatak uvjeta da sprema samo koordinate loše orijentiranih komponenti.

Zadnje što je preostalo je pozvati funkcije, što je prikazano kodom na Slika 87.

```
107 if __name__ == '__main__':
108     while True:
109         # analiza i izlaz iz programa
110         if keyboard.is_pressed('esc'):
111             camera.stop_preview()
112             print('camera stop')
113             sleep(0.5)
114             print("Analiza!")
115             gkod()
116             print('g done')
117             print('Exit!')
118             break
119         # procesiranje slika
120         elif keyboard.is_pressed('p'):
121           sleep(0.5)
122           camera.capture(path)
123           camera.capture_sequence([path2 %br])
124           camera.capture_sequence([path3 %br])
125           sleep(2)
126           print('image captured')
127           obrada()
128           br += 1 # za pojedinu sliku
129         elif keyboard.is_pressed('z'):
130           camera.stop_preview()
131           sleep(2)
132           break
```

Slika 87 Raspberry Pi pozivanje funkcija

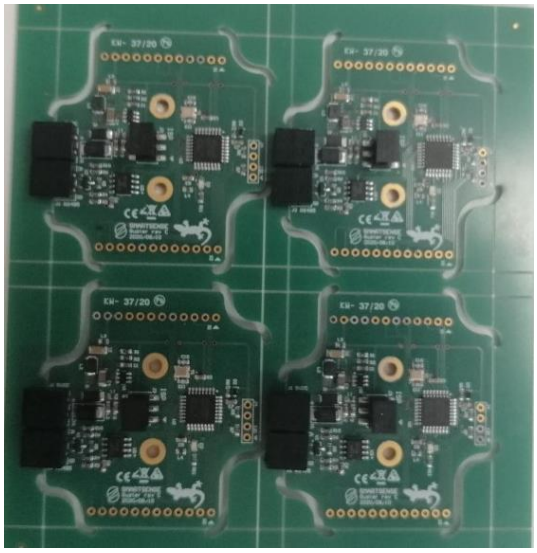
- Pozicioniranje na netočne komponente G-kodom

Kao i kod prethodno opisanog Pozicioniranja G-kodom. Dobiveni G-kod za pozicioniranje na netočne komponente se ubacuje u UCCNC program. Pokretanjem se kamera pozicionira točno iznad predviđene loše komponente.

Ovaj proces je napravljen da se olakša pronalaženje loše okrenute komponente ukoliko se na PCB panelu nalazi veliki broj komponenata od interesa. Praksa je pokazala kako se prilikom ručne inspekcije lako preskoči pokoja komponenta, čime se smanjuje šansa za uočavanje i ispravljanje greške.

- Rezultati:

Prvi test izrađenog sustava napravljen je nad PCB panelom, na kojem postoji osam komponenata od interesa (dvije komponente puta 4 PCB-a, Slika 88).



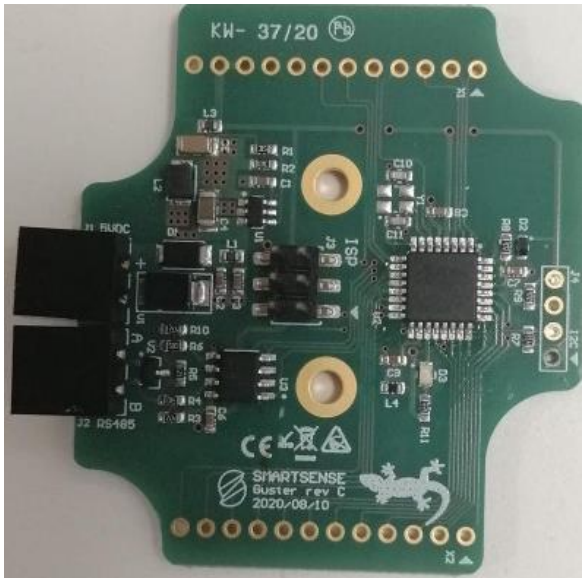
Slika 88. PCB panel za testiranje 1

Prethodnom provjerom je potvrđeno kako su na panelu sve komponente pravilne orijentacije. Rezultat je bio pozitivan. Algoritam je točno pogodio sve komponente .

```
['GOOD', 'GOOD', 'GOOD', 'GOOD', 'GOOD', 'GOOD', 'GOOD', 'GOOD']
```

Slika 89. Predviđanje 1 AOI sustava

Kako bi se potvrdili dobro dobiveni rezultati, testiranje je ponovljeno na jednom ne zalemljenom PCB-u na kojem je moguće namjestiti lošu orijentaciju komponente (Slika 90).



Slika 90. PCB za testiranje 2

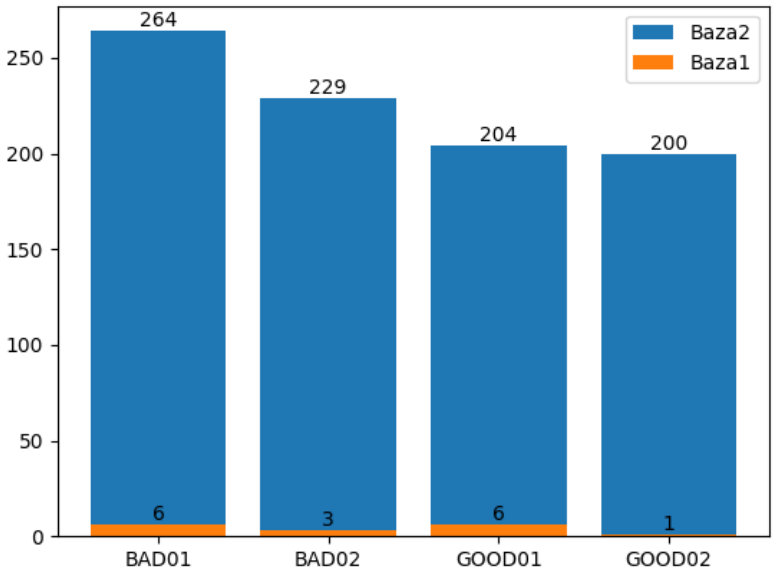
Kao i prilikom prethodnog testiranja i tu je potvrđena orijentacija komponenti. U ovom slučaju obje komponente su krive orijentacije. Suprotno očekivanom, rezultati za ovaj slučaj ispadaju negativni, tj. algoritam komponente loše orijentacije smješta u kategoriju komponenti dobre orijentacije (Slika 91).

```
['GOOD', 'GOOD']
```

Slika 91. Predviđanje 2 AOI sustava

Na ova dva testna primjera se vidi kako je prethodno u radu krivo određeno da je Algoritam 4 dobar, iako se analizom konfuzijske matrice pokazalo kako se radi o najboljem izboru. No, zašto je to tako? Ako se pažljivo promotri Slika 58 može se vidjeti kako se većina originalnih podataka nalazi u kategorijama dobro orijentiranih komponenti i to čak 72,7%. Ukoliko se to usporedi s testnim podacima za konfuzijsku matricu, može se vidjeti kako se unutar testnih podataka loše orijentiranih komponenti nalaze samo slike obrađene augmentacijom. Iz navedenih činjenica vidi se da je algoritam naučio da sve originalne slike pripadaju kategoriji komponenti s dobrom orijentacijom.

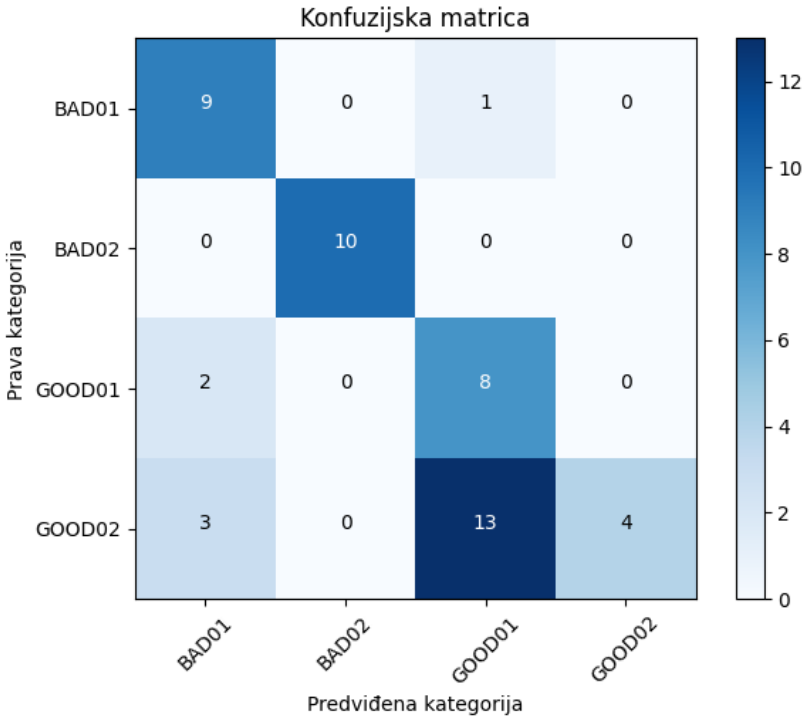
Kako bi se navedeno potvrdilo izrađen je novi algoritam, Algoritam 6. Za navedeni algoritam je napravljena nova baza podataka na način da se iz originalne baze (Baza 1, Slika 58) izbace slike na način da većina bude u korist kategorija loše okrenutih komponenti (Slika 92).



Slika 92. Graf raspodjele podataka u bazi podataka 2 prije augmentacije (Baza1) i poslije (Baza2)

Tablica 12. Parametri Algoritma 6

Baza podataka	Baza2 (Slika 92)
Veličina serije slika za trening	20
Veličina serije slika za validaciju	4
Dubina sloja učenja	-23
Stopa učenja	0.00025
Epohe	50



Slika 93. Algoritam 6: Konfuzijska matrica

Usporedbom konfuzijske matrica ovog algoritma sa konfuzijskom matricom prethodno odabranog Algoritma 3 (Slika 72), vidi se kako je trenutni algoritam inferioran sa 95% točnošću predviđanja komponenti loše orijentacije te 83,3% točnošću predviđanja komponenti dobre orijentacije.

Prilikom testiranja AOI sustava s Algoritmom 6 na prethodnim primjerima (Slika 88 i Slika 90) dobije se da algoritam i dalje ne radi dobro. U slučaju PCB panela sa svim dobro okrenutim komponentama on pogađa kako su sve loše okrenute (Slika 94), čime mu je točnost 0%.

```
['BAD', 'BAD', 'BAD', 'BAD', 'BAD', 'BAD', 'BAD', 'BAD']
```

Slika 94. Predviđanje 3 AOI sustava

U slučaju PCB pločice algoritam pogađa sa 50% točnošću (Slika 95), što i dalje ne zadovoljava cilj rada AOI sustava.

```
['GOOD', 'BAD']
```

Slika 95. Predviđanje 4 AOI sustava

Ovim rezultatima se može potvrditi teza kako je problem bio nejednakost podataka, no zbog manjka podataka se i dalje ne može dobiti adekvatan algoritam.

5. ZAKLJUČAK

Zadatak ovog diplomskog rada je bio osmisliti SMT proizvodnu liniju s AOI sustavom, analizirati probleme i greške SMT proizvodnje te osmisliti vlastiti algoritam za AOI i implementirati ga.

Izradom 3D modela pojedine komponente SMT proizvodnje i analizom najčešćih grešaka, adekvatno se odradilo ubacivanje AOI sustava iza *pick and place* sustava kako bi se detektirale greške krive orijentacije komponenti. Osmišljeni sustav se poklapa s dimenzijama i funkcionalnostima prostora stoga ga nije problem implementirati.

Tehnikama *data augmentation* i *transfer learning* se nije uspio kompenzirati manjak podataka u bazi podataka za razvoj adekvatnog algoritma.

Jedan od problema koji se javio je nepredvidivost augmentacije u smislu da se ne može svaki podatak u bazi podataka obraditi na isti način. Rezultat toga je da je mreža učila krive karakteristike sa slike, gdje se prilikom drugog testiranja moglo vidjeti kako je mreža imala 0% učinkovitosti. Analiza tog problema je rezultirala zaključkom da je mreža istrenirana da, ako se na ulazu nalazi originalna (ne augmentirana) slika, zaključi da se radi o dobro okrenutoj komponenti. Razlog tome je znatno veći udio originalnih podataka komponenti dobre orijentacije u bazi podataka, u odnosu na one loše orijentacije, čime mreža zaključuje da će pogađanje kategorije dobro okrenutih rezultirati većom točnošću. Isto se kasnije potvrdilo smanjenjem baze podataka za dobro orijentirane komponente, no to je rezultiralo daljnjim smanjenjem podataka unutar baze i lošim rezultatima testiranja algoritma. Rješenje ovog problema bi bilo skupljanje većeg seta podataka jednakih dimenzija.

Drugi problem je u *transfer learning*-u gdje se kompleksnu mrežu koja ima mogućnost detekcije kompleksnih karakteristika, poput uha, nosa, kotača, krila itd., pokušava dotrenirati da uočava jednostavne greške orijentacije. Uspješnije bi bilo uzeti slojeve mreže koji su trenirani za detekciju bitnih karakteristika za novi algoritam, ostale eliminirati, te dodati vlastite slojeve koji bi odradili bitnu kategorizaciju. Vizualizaciju pojedinog sloja mreže je moguće napraviti aplikacijom razvijenom u [33].

Implementacija algoritma, tj. izrada AOI sustava, izuzev algoritma, je uspješno odrađena kombinacijom UCCNC programa za kontrolu CNC stroja, te Raspberry Pi računala i Raspberry Pi kamere za uzimanje i obradu slike.

LITERATURA

[1] SMT, https://www.surfacemountprocess.com/, 15.06.2021.

[2] Altium designer, https://www.altium.com/altium-designer/, 15.06.2021.

[3] EAGLE, https://www.autodesk.com/products/eagle/overview?term=1-YEAR, 15.06.2021.

[4] *Reflow* lemljenje, https://hr.melayukini.net/wiki/Reflow_soldering, 16.06.2021.

[5] *Gerber* datoteke, https://www.vse.com/blog/2019/10/29/gerber-files-explained-understanding-their-role-in-pcb-manufacturing/, 16.06.2021.

[6] Squeegee, https://en.wikipedia.org/wiki/Squeegee, 16.06.2021.

[7] *Jet printing*, https://www.techbriefs.com/component/content/article/tb/pub/techbriefs/manufacturing-prototyping/23681, 16.06.2021.

[8] Selektivno lemljenje, https://en.wikipedia.org/wiki/Selective_soldering, 16.06.2021.

[9] Valno lemljenje, https://en.wikipedia.org/wiki/Wave_soldering, 16.06.2021.

[10] Usporedba selektivnog i valnog lemljenja, https://www.technotronix.us/pcbblog/selective-versus-wave-soldering-which-one-is-better-for-pcb-assembly/, 16.06.2021.

[11] RTG inspekcija, https://en.wikipedia.org/wiki/Automated_X-ray_inspection, 16.06.2021.

[12] Liao , H., Lim, Z., Hu, Y. i Tseng, H., „Guidelines of Automated Optical Inspection (AOI) System Development“, 3rd International Conference on Signal and Image Processing (ICSIP), 2018., pp. 362-366, doi: 10.1109/SIPROCESS.2018.8600456.

[13] Jain , R., Kasturi, R. i Schunck, B. G., „Machine vision“, McGraw-Hill, Inc., New York, USA, 1995.

[14] Sonka, M., Hlavac, V. i Boyle, R., „Image Processing, Analysis, and Machine Vision“, Cengage Learning Inc., Stamford, USA, 2013.

[15] Uvod u strojni vid, https://www.assemblymag.com/ext/resources/White_Papers/Sep16/Introduction-to-Machine-Vision.pdf, 17.06.2021.

[16] Niaz , M., Klassen, S., McMillan B. i Metz, D. „Reconstruction of the history of the photoelectric effect and its implications for general physics textbooks“, Wiley Periodicals, Inc., 2010., pp. 1-29, doi: 10.1002/sce.20389.

[17] Senzori slike, https://www.fer.unizg.hr/_download/repository/Senzori_Slike-171208-FER.pdf, 18.06.2021.

[18] Green , M. E., „Photovoltaic principles, Physica E: Low-dimensional Systems and Nanostructures“, Elsevier, Sydney, Australia, 2002.

[19] S. Cubero , S., Aleixos, N., Moltó, E., Gómez-Sanchis, J. i Blasco, J., "Advances in machine vision applications for automatic inspection and quality evaluation of fruits and vegetables," Food and Bioprocess Technology, Vol. 4, No. 4, 2011., pp. 487-504.

[20] AOI tehnologija, https://www.market-prospects.com/articles/what-is-aoi-technology, 16.06.2021.

[21] Mohri , M., Rostamizadeh, A. i Talwalkar, A., "Foundations of Machine Learning", The MIT Press, Cambridge, MA, 2018.

[22] Novaković , B., Majetić, D. i Široki, M., „Umjetne neuronske mreže. = Artificial neural networks“ Sveučilište u Zagrebu, Fakultet strojarstva i brodogradnje, Zagreb, Hrvatska, 1998., ISBN 953-6313-17-0.

[23] Albawi, S., Mohammed, T. A., i Al-Zawi, S., „Understanding of a convolutional neural network“, 2017 International Conference on Engineering and Technology (ICET), 2017., pp. 1-6, doi:10.1109/icengtechnol.2017.8308186.

[24] Ekra serio 4000, http://www.amtest-smt.com/hr/proizvodi/oprema-za-proizvodnju/printanje-lemne-paste/ekra/ekra-serio-4000-back-to-back,489.html, 22.06.2021.

[25] Hanwha SM482 Plus, http://www.amtest-smt.com/hr/proizvodi/oprema-za-proizvodnju/polaganje-smt-komponenti/hanwha-precision-machinery/hanwha-sm482-plus,497.html, 22.06.2021.

[26] HELLER 1707 MK 5, http://www.amtest-smt.com/hr/proizvodi/oprema-za-proizvodnju/peci-za-lemljenje-i-susenje/protocno-lemljenje/heller-1707-mk5-series-smt-reflow-system,431.html, 22.06.2021.

[27] Premošćivanje lemom, https://www.allaboutcircuits.com/technical-articles/a-solder-bridge-to-nowhere-what-is-a-solder-bridge-how-to-prevent/, 23.06.2021.

[28] Fiziča odvojenost vodića i nožice, https://www.seeedstudio.com/blog/2019/08/07/13-common-pcb-soldering-problems-to-avoid/, 23.06.2021.

[29] PARMI Exceed, http://www.amtest-smt.com/en/products/inspection-and-test-equipment/automatic-optical-inspection/parmi/parmi-xceed,352.html, 23.06.2021.

[30] Raspberry Pi High Quality Camera: https://www.raspberrypi.org/products/raspberry-pi-high-quality-camera/, 24.06.2021.

[31] Raspberry Pi 3 specifikacije, https://www.raspberrypi.org/products/raspberry-pi-3-model-b/, 24.06.2021.

[32] Stepcraft-2/D.420: https://shop.stepcraft-systems.com/stepcraft-2-420-construction-kit, 24.06.2021.

[33] Vlahović, S., "Vizualizacija rada konvolucijske neuronske mreže", Završni rad, Sveučilište u Zagrebu, Fakultet strojarstva i brodogradnje, Zagreb, 2020. Dostupno na: https://urn.nsk.hr/urn:nbn:hr:235:108019

PRILOZI

I. Python kod

II. Tehnička dokumentacija

I Python

 # Algoritam_kamera

```
from picamera import PiCamera
from time import sleep
from PIL import Image
import keyboard

br = 1
camera = PiCamera()
camera.resolution = (300, 300)
camera.start_preview()
img = Image.open('Slike/kriz2.png')

pad = Image.new('RGB', (
    ((img.size[0] + 31) // 32) * 32,
    ((img.size[1] + 15) // 16) * 16,
))
pad.paste(img, (0, 0))
o = camera.add_overlay(pad.tobytes(), size=img.size)

o.alpha = 130
o.layer = 3

while True:
    if keyboard.is_pressed('esc'):
        print('Exit!')
        camera.stop_preview()
        break
```

```
    elif keyboard.is_pressed('p'):
        print('Take pitcure')
        camera.capture_sequence(['/home/pi/Desktop/DIPLOMSKI/Slike/01/%02d.jpg' % br])
        sleep(0.5)
        br += 1
```

 # Algoritam_slike

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import shutil
import numpy as np
import matplotlib.pyplot as plt
import os
import random

IM_SIZE = 224

# Data Augmentation
gen = ImageDataGenerator(rotation_range=10,
                         width_shift_range=0.1,
                         height_shift_range=0.1,
                         shear_range=0.15,
                         zoom_range=0.1,
                         channel_shift_range=10.,
                         horizontal_flip=True
                         )

PATH = 'Slike/Data5'
os.chdir(PATH)
```

```
for i in os.listdir():
    br = 1
    size = int(300 / len(os.listdir(f'{i}')))
    for j in os.listdir(f'{i}'):
        image = np.expand_dims(plt.imread(i + '/' + j), 0)


        if i == 'GOOD01' or i == 'GOOD02':
            for k in range(size):
                p = "AI-" + str(br) + "-"
                aug_iter = gen.flow(image,
                                    batch_size=10,
                                    save_to_dir=i,
                                    save_prefix=p,
                                    save_format='jpg',
                                    )
                aug_iter.next()
        else:
            for k in range(size+10):
                p = "AI-" + str(br) + "-"
                aug_iter = gen.flow(image,
                                    batch_size=10,
                                    save_to_dir=i,
                                    save_prefix=p,
                                    save_format='jpg',
                                    )
                aug_iter.next()
        br += 1
```

```
# Priprema DATASET-a
CATEGORIES = ['BAD01', 'BAD02', 'GOOD01', 'GOOD02']

if os.path.isdir('train/0/') is False:
    os.mkdir('train')
    os.mkdir('valid')
    os.mkdir('test')

    for i in CATEGORIES:
        shutil.move(f'{i}', 'train')
        os.mkdir(f'valid/{i}')
        os.mkdir(f'test/{i}')

        valid_samples = random.sample(os.listdir(f'train/{i}'), 40)
        for j in valid_samples:
            shutil.move(f'train/{i}/{j}', f'valid/{i}')

        test_samples = random.sample(os.listdir(f'train/{i}'), 10)
        for k in test_samples:
            shutil.move(f'train/{i}/{k}', f'test/{i}')
os.chdir('../..')

train_path = 'Slike/Data2/train'
valid_path = 'Slike/Data2/valid'
test_path = 'Slike/Data2/test'

# Algoritam
import tensorflow as tf
from tensorflow.keras.datasets import cifar10
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten
from tensorflow.keras.layers import Conv2D, MaxPooling2D
from tensorflow.keras.callbacks import TensorBoard
import numpy as np

import cv2.cv2 as cv2

import pickle
import time

train_path = 'Slike/Data5/train'
valid_path = 'Slike/Data5/valid'

IM_SIZE = 224

train_batches = ImageDataGenerator(

preprocessing_function=tf.keras.applications.resnet50.preprocess_input).flow_from_directory
(
    directory=train_path,
    target_size=(IM_SIZE, IM_SIZE),
    batch_size=10
)
valid_batches = ImageDataGenerator(

preprocessing_function=tf.keras.applications.resnet50.preprocess_input).flow_from_directory
(
```

```
    directory=valid_path,
    target_size=(IM_SIZE, IM_SIZE),
    batch_size=5
)

NAME = 'Testna-mreza-v06-{}'.format(int(time.time()))

tensorboard = TensorBoard(log_dir='logs_07/{}'.format(NAME))

# Funkcije zaustavljanja učenja mreže
class StopOnPoint(tf.keras.callbacks.Callback):
    def __init__(self, point):
        super(StopOnPoint, self).__init__()
        self.point = point

    def on_epoch_end(self, epoch, logs=None):
        accuracy = logs["accuracy"]
        accuracy_v = logs["val_accuracy"]
        if accuracy >= self.point and accuracy_v >=self.point:
            self.model.stop_training = True



# Odabir CNN-a
res = tf.keras.applications.resnet50.ResNet50()

# Stvaranje novog modela
x = res.layers[-1].output
output = Dense(units=4, activation='softmax')(x)
model = tf.keras.Model(inputs=res.input, outputs=output)
```

```
# Određivanje slojeva koji se mogu trenirati
for layer in model.layers[:-39]:
    layer.trainable = False

# Karakteristike treniranja
model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)

# Pozivanje funkcije za zaustavljanje mreže u posebnim uvjetima
callbacks = [StopOnPoint(1.0)]

# Treniranje i spremanje mreže
model.fit(x=train_batches, validation_data=valid_batches, epochs=50,
callbacks=[tensorboard, callbacks])
model.save('20210617_testni_00.model')
```

 # Algoritam_testiranje

```
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import numpy as np
import itertools
from sklearn.metrics import confusion_matrix, plot_confusion_matrix
import tensorflow as tf
```

```
def plot_cm(cm, classes,
        normalize=False,
        title='Konfuzijska matrica',
        cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")
```

```
    plt.tight_layout()
    plt.ylabel('Prava kategorija')
    plt.xlabel('Predviđena kategorija')



model = tf.keras.models.load_model('20210627_testni_08.model')

test_path = 'Slike/Data5/test/'
IM_SIZE = 224

test_batches = ImageDataGenerator(

preprocessing_function=tf.keras.applications.resnet50.preprocess_input).flow_from_directory
(
    directory=test_path,
    target_size=(IM_SIZE, IM_SIZE),
    batch_size=1,
    shuffle=False
)

test_labels = test_batches.classes
predictions = model.predict(x=test_batches, verbose=0)
cm = confusion_matrix(y_true=test_labels, y_pred=predictions.argmax(axis=1))
cm_plot_labels = ['BAD01', 'BAD02', 'GOOD01', 'GOOD02']
plot_cm(cm=cm, classes=cm_plot_labels, title='Konfuzijska matrica')
plt.show()
```

 # Pretvaranje u G-kod

```
import pandas as pd
```

```
import os, sys

lista = {}
lista2 = []
# dezignirana mjesta komponenti od interesa
komponente = ['U2', 'U3']
br = 0

# pozicioniranje u datoteku unutar koje će se raditi obrada
os.chdir('P&P_data')

# otvaranje excel datoteke
df = pd.read_excel('PickAndPlace_Guster 1.xlsx')

# kreiranje g-kod datoteke
fd = os.open('P&P_Guster.nc', os.O_RDWR|os.O_CREAT)

# kreiranje početnih komamndi g-koda
# G90-apsolutna distanca, G0-linearna interpolacija sa maksimalnom brzinom,
# G53-linearna interpolacija u koordinatnom sustavu stroja
# G64-kontrola putanje po konstantnoj brzini
# G54-work offset
os.write(fd, bytes('G90\nG0 G53 Z0.\n\nG64\nG54\nZ0.',
                   encoding='utf8'))

# definiranje odstupanja za panel
array = [0, 0, -56, 0, 0, 62, -56, 62]

# izvlačenje koordinata iz excel datoteke i prebacivanje u g-kod
```

```
for j in range(int(len(array)/2)):
    for i in range(len(df.values)):
        if df.values[i][0] in komponente:
            # vrijednost y koordinate
            Y_val = str(float(df.values[i][1]*-1 + array[br*2+1]))
            # vrijednost x koordinate
            X_val = str(float(df.values[i][2] + array[br*2]))
            # kreiranje g-koda
            X = bytes(("\nY" + Y_val + " X" + X_val + "\nM0"),
                      encoding='utf8')
            os.write(fd, X)
    br += 1
# završavanje g-koda
os.write(fd, bytes('\nM30',
                   encoding='utf8'))
```

 # AOI_kod

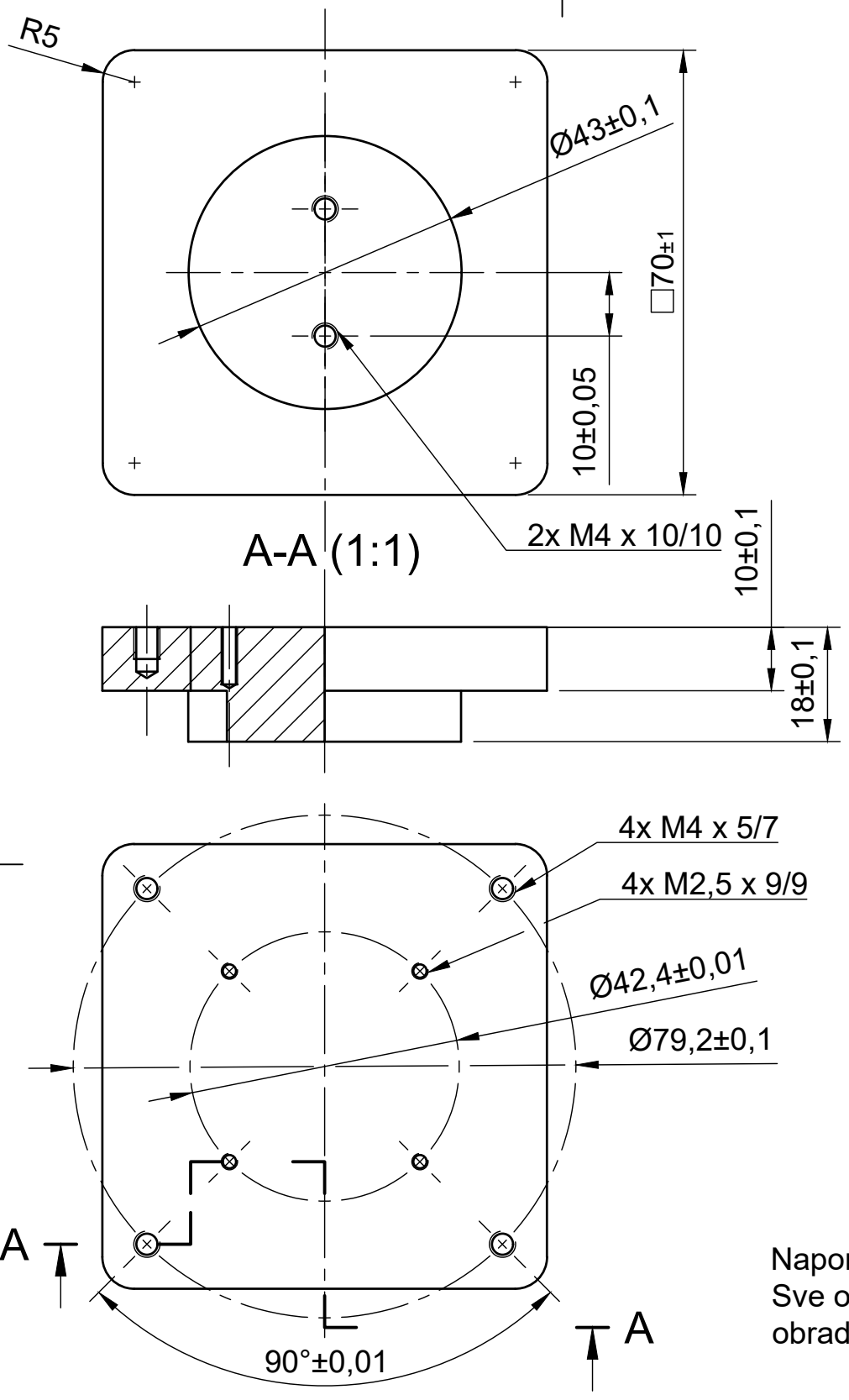
```
from picamera import PiCamera
from time import sleep
from PIL import Image, ImageDraw, ImageFont
import keyboard
import numpy as np

import tensorflow as tf

import pandas as pd
import os, sys

import multiprocessing
```

```
model = tf.keras.models.load_model('20210627_testni_04.model')

path = '/home/pi/Desktop/DIPLOMSKI/Slike/Test2/01.jpg'
path2 = '/home/pi/Desktop/DIPLOMSKI/Slike/Analiza/%02d.jpg'
IM_SIZE = 224
KATEGORIJE = ['GOOD', 'BAD']
BAD = [0, 1]
predikcija = []
br = 1
broj = 0
br2 = 0
font = ImageFont.truetype("/home/pi/Fonts/EUROCAPS.TTF", size=50)

lista = {}

komponente = ['U2', 'U3']
array = [0, 0]

camera = PiCamera()
camera.resolution = (225, 225)
camera.start_preview()
img = Image.open('Slike/kriz2.png')
sleep(3.0)

pad = Image.new('RGB', (
    ((img.size[0] + 31) // 32) * 32,
    ((img.size[1] + 15) // 16) * 16,
))
```

```
pad.paste(img, (0, 0))
o = camera.add_overlay(pad.tobytes(),
                       size=img.size
                       )

o.alpha = 130
o.layer = 3



# Procesiranje: path, IM_SIZE, br, camera
def obrada():
    slika = tf.keras.preprocessing.image.load_img(path, target_size=(IM_SIZE, IM_SIZE))
    array1 = tf.keras.preprocessing.image.img_to_array(slika)
    array1 = np.array([array1])
    predictions = model.predict(array1)[0]
    for i in range(len(predictions)):
        if predictions[i] == max(predictions):
            if i in BAD:
                predikcija.append(KATEGORIJE[1])
                img = Image.open(path2 %br)
                ImageDraw.Draw(img).text((20, 20), KATEGORIJE[1], fill=(255, 0, 0), font=font)
                img.save(path2 %br)
            else:
                predikcija.append(KATEGORIJE[0])
                img = Image.open(path2 %br)
                ImageDraw.Draw(img).text((20, 20), KATEGORIJE[0], fill=(0, 255, 0), font=font)
                img.save(path2 %br)
    print('gotovo')
    return predikcija
```

```
# Generiranje pokaznog koda za loše komponente
def gkod():
    os.chdir('P&P_data')
    fd = os.open('P&P_Guster_BAD.nc', os.O_RDWR | os.O_CREAT)
    os.write(fd, bytes('G90\nG0 G53 Z0.\n\nG64\nG54\nZ0.',
                       encoding='utf8'))
    df = pd.read_excel('PickAndPlace_Guster 1.xlsx')
    broj = 0
    for j in range(int(len(predikcija)/len(komponente))):  # stvaranje arraya
        br2 = 0
        for i in range(len(df.values)):  # Ide po svim vrijednostima iz excela
            if df.values[i][0] in komponente:
                if predikcija[br2] == KATEGORIJE[1]:  # ako je komponenta iz kategorije
komponente i ako je predikcija loša
                    X = bytes(("\nY" + str(float(df.values[i][1]*-1 + array[broj * 2 + 1])) + " X" +
str(
                        float(df.values[i][2] + array[broj * 2])) + "\nM0"), encoding='utf8')
                    os.write(fd, X)
                br2 += 1
        broj += 1  # broji array ide do 4
    os.write(fd, bytes('\nM30', encoding='utf8'))


if __name__ == '__main__':
    while True:
        print('\n\nSTART\n\n')
        # analiza i izlaz iz programa
        if keyboard.is_pressed('esc'):
```

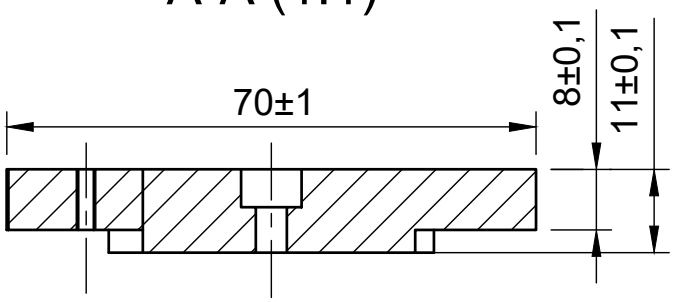
```
        camera.stop_preview()
        print('camera stop')
        sleep(0.5)

        print("Analiza!")
        gkod()
        print('g done')
        print('Exit!')
        break
    # procesiranje slika
    elif keyboard.is_pressed('p'):
      sleep(0.5)
      camera.capture(path)
      camera.capture_sequence([path2 %br])
      sleep(2)
      print('image captured')
      obrada()
      br += 1 # za pojedinu sliku
    elif keyboard.is_pressed('z'):
      camera.stop_preview()
      sleep(2)
      break
```

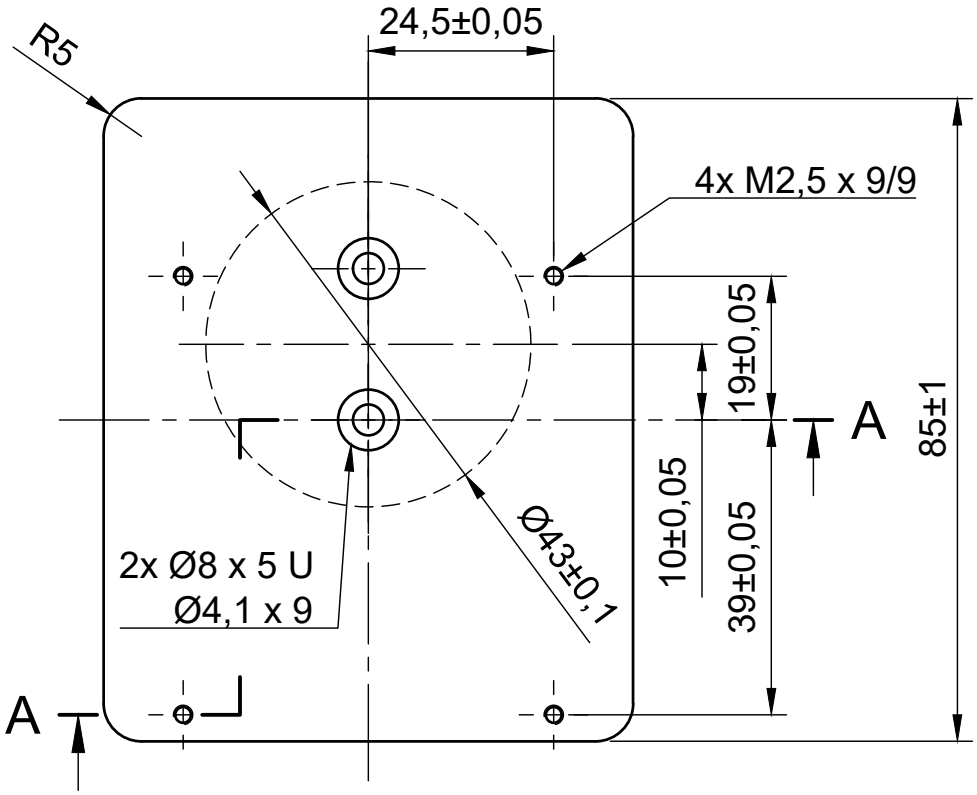


Napomena:
Sve oštre rubove obraditi na 1x45°

Pozicija:	Mjerilo originala:	Projektirao, razradio i crtao:	Potpis:	
	1:1	Stjepan Vlahović 25.6.2021.		
		Materijal:	Masa:	DIPLOMSKI RAD
		EN AW 2017	0,138 kg	
		Naziv:	Broj crteža:	
		Adapter_01	202106250001	
			Rev.: 01 / Format: A4 / Listova: 1 / List: 1/1	

A-A (1:1)





Napomena:
Sve oštre rubove obraditi
1X45°

Pozicija:	Mjerilo originala: 1:1	Projektirao, razradio i crtao: Stjepan Vlahović 25.6.2021.	Potpis:	
		Materijal: EN AW 2017	Masa: 0,162 kg	DIPLOMSKI RAD
		Naslov: Adapter_02	Broj crteža: 202106250002	
			Rev.: 01 / Format: A4 / Listovi: 1 / List: 1/1	