

SVEUČILIŠTE U ZAGREBU  
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 5177

**Lokalizacija i očitavanja rukom pisanih  
bodova studenta**

Domagoj Plušćec

Zagreb, lipanj 2017.

Zahvaljujem mentoru doc. dr. sc. Marku Čupiću, svojoj obitelji te svim nastavnicima Fakulteta elektrotehnike i računarstva u Zagrebu koji su mi pomogli tijekom studija te time na neki način doprinijeli i ovom radu.

## Sadržaj

1.	Uvod .....	1
2.	Pregled problema.....	2
2.1.	Obrada slike.....	4
2.1.1.	Pretvaranje boja u nijanse sive .....	4
2.1.2.	Binarizacija slike .....	6
2.1.3.	Određivanje orijentacije i veličine obrasca .....	9
2.2.	Segmentacija znamenki .....	12
2.3.	Klasifikacija znamenki .....	17
2.3.1.	Model umjetne neuronske mreže.....	17
2.3.2.	Učenje umjetne neuronske mreže.....	20
2.3.3.	Učenje algoritmom propagacije pogreške unatrag .....	22
3.	Priprema skeniranog obrasca.....	26
3.1.	Definiranje obrasca.....	26
3.2.	Odbacivanje suvišnih informacija .....	32
3.3.	Određivanje orijentacije slike i izdvajanje polja s bodovima studenata.....	35
4.	Segmentacija znamenki .....	39
4.1.	Algoritam označavanja povezanih komponenti .....	39
4.2.	Filtriranje segmenata .....	41
4.3.	Rezultati segmentacije .....	44
5.	Klasifikacija znamenki .....	48
5.1.	Prilagodba ulaznih podataka.....	48
5.2.	Arhitektura neuronske mreže.....	50
5.3.	Rezultati klasifikacije .....	51
6.	Implementacija sustava.....	53
6.1.1.	Grafičko korisničko sučelje .....	53

6.2.	Primjer izlazne datoteke .....	57
6.3.	Raspored komponenata unutar programskih paketa.....	57
7.	Rezultati obrade obrazaca.....	61
8.	Zaključak .....	64
	Literatura .....	65
	Sažetak.....	67
	Summary.....	68

## Popis slika

Slika 1. Uokvireni predložak obrasca.....	2
Slika 2. Pregled procesa obrade skeniranog obrasca .....	3
Slika 3. pristupi segmentaciji (prilagođeno iz [7]) .....	12
Slika 4. Primjer slike broja s odvojenim znamenkama .....	13
Slika 5. Primjer slike broja s prekrivenim znamenkama .....	14
Slika 6. Primjer slike broja sa spojenim znamenkama .....	15
Slika 7. Primjer segmentacije slike 4. algoritmom povezanih komponenti .....	16
Slika 8. Primjer segmentacije slike 5. algoritmom povezanih komponenti .....	16
Slika 9. Primjer segmentacije slike 6. algoritmom povezanih komponenti .....	16
Slika 10. Biološki neuron (prilagođeno iz [14]) .....	17
Slika 11. Model umjetnog neurona (preuzeto iz [14]) .....	18
Slika 12. Slojevit neuronska mreža arhitekture $2 \times 3 \times 3 \times 2$ (preuzeto iz [14]) .....	20
Slika 13. Određivanje granice učenja (preuzeto iz [14]) .....	21
Slika 14. Granice regija dobivene algoritmom 5 .....	31
Slika 15. Usporedba uokvirenog predloška obrasca i uokvirenog skeniranog obrasca.....	36
Slika 16. Primjer izrezane regije.....	38
Slika 17. Primjer greške prilikom izrezivanja regije s bodovima studenata .....	38
Slika 18. Prikaz susjedstva od 8 slikovnih elemenata .....	39
Slika 19. Susjedstvo od 8 slikovnih elemenata s označenim susjedima.....	41
Slika 20. Prikaz horizontalne projekcije te gornje i donje osnovne linije (preuzeto iz [21]) .....	42
Slika 21. Primjer normalizacije slika segmenata .....	48
Slika 22. Primjer krivo klasificiranih znamenki. 8 je klasificirano kao 6, 9 je klasificirano kao 3 i 1 je krivo klasificirano kao decimalni separator .....	52
Slika 23. Prikaz praznog prikaza za definiciju predloška obrasca .....	53
Slika 24. Prikaz prozora za definiciju predloška obrasca s definiranim obrascem .....	54

Slika 25. Prikaz za analizu obrasca s transformiranim obrascem .....	55
Slika 26. Prozor s ćelijom s bodovima studenta te prozor sa segmentom broja.....	55
Slika 27. Prikaz za treniranje neuronske mreže.....	56
Slika 28. Prikaz za obradu skeniranih obrazaca .....	56
Slika 29. Primjer djela izlazne datoteke .....	57
Slika 30. Opći raspored paketa .....	58
Slika 31. Prikaz paketa form.....	58
Slika 32. Prikaz paketa image .....	59
Slika 33. Prikaz paketa neural .....	60
Slika 34. Prikaz paketa studentforms .....	60

# 1. Uvod

U današnjem obrazovnom sustavu nailazimo na ispite koje polaže veliki broj ljudi. Primjer su ispiti na fakultetskim predmetima koji mogu imati i nekoliko stotina studenata koji polažu ispite te državna matura koju piše nekoliko tisuća učenika srednjih škola svake godine.

Na ispitima koje polaže veliki broj ljudi pokušava se uvesti ispite bazirane na ponuđenim odgovorima (engl. *multiple choice questions*) koji se mogu automatski ispravljati [1]. Međutim, zadatke s ponuđenim odgovorima nije moguće koristiti kod nekih tipova zadataka (npr. kada je potrebno nacrtati shemu digitalnog sklopa) i kada želimo bodovati postupak rješavanja zadatka.

Nakon što ispravljajući isprave ispite potrebno je bodove svakog ispita unijeti u računalni sustav. Takav proces uzima puno vremena te je sklon greškama. Problemom automatizacije unosa bodova studenata su se bavili i na sveučilištu „King Mongkut's University of Technology North Bangkok“, Tajland gdje je napravljen sustav za prepoznavanje podataka unesenih na ispitnu košuljicu [2]. U okviru ovog završnog rada bit će razvijen sustav za lokalizaciju i očitavanja rukom pisanih bodova studenata kojim će biti moguće automatizirano unositi bodove studenata u računalni sustav.

Sustav i procesi unutar sustava koje problem obuhvaća opisani su u drugom poglavlju. U trećem poglavlju opisan je postupak pripreme obrasca gdje se skenirani obrazac obrađuje te se iz njega izdvajaju regije koje sadrže bodove studenata. Nakon izdvajanja regija bodovi se segmentiraju na znamenke, a proces segmentacije implementiran u sklopu ovog rada dan je u četvrtom poglavlju. U petom poglavlju opisan je postupak klasifikacije pojedinih znamenki. U ostalim poglavljima opisana je implementacija sustava te rezultati koje sustav postiže.

## 2. Pregled problema

Kako bi u potpunosti definirali problem potrebno je definirati obrazac u koji će se unositi rukom pisani bodovi studenata. Obrazac korišten u sklopu ovog rada prikazan je na slici 1.

Dijelovi obrasca su tablica koja sadrži polja s očekivanim unosom, prazna polja u koje se unosi rukom pisani iznos bodova te tri crna kvadratna markera koji će biti korišteni prilikom analize obrasca.

0		8.9		24549	
1		9.0		23061	
2		13.14		6059	
3		70.97		2058	
4		0.501		3,03	
5		1.767		8,07	
6		2.818		1,08	
7		3.484		9,09	
8		4.955		0,24	
9		5.152		6,33	
12		6.99		2,0	
34		7.771		4,38	
56		8.835		5,96	
78		9.985		7,90	
90		10.005		8,6	
31		16.219		5,43	
42		446.17		10,22	
53		89.626		3,45	
64		28.65		6,8	
75		73869		12,3	
0.1		727.802		40,5	
1.2		4041		9,88	
2.3		4374		7,65	
3.4		7922		14,73	
4.5		33291		92,60	
5.6		8257		0,636	
6.7		6687		11,96	
7.8		93947		51,88	

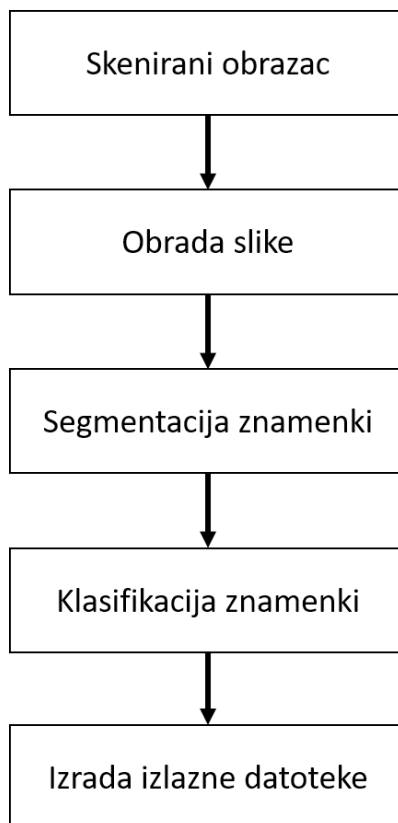
Slika 1. Uokvireni predložak obrasca



Proces obrade obrasca se sastoji od sljedećih faza.

- Skeniranje obrazaca. Svi obrasci su skenirani u rezoluciji od 200 i 300 DPI (engl. *dots per inch*) te spremljeni u png formatu.
- Obrada slike obrazaca. Postupak obrade slike je detaljno opisan u poglavlju 2.1.
- Segmentacija znamenki. Pregled pristupa segmentaciji znamenki dan je u poglavlju 2.2.
- Klasifikacija znamenki. Teoretski pregled odabranog pristupa klasifikaciji dan je u poglavlju 2.3.
- Stvaranje izlazne datoteke. Nakon klasifikacije pojedinih znamenki potrebno je napraviti datoteku koja sadrži listu detektiranih bodova na obrascu te po potrebi oznaku ako je sustav ustanovio da ne može klasificirati barem jedan od unesenih bodova kako bi se bodovi s takvog obrasca mogli naknadno ručno unijeti u sustav koji bilježi bodove.

Pregled procesa obrade skeniranog obrasca dan je na slici 2. Proces obrade obrasca temeljen je na postupku obrade koji je korišten u [2].



Slika 2. Pregled procesa obrade skeniranog obrasca

## 2.1. Obrada slike

Nakon što smo skenirali obrazac dobivamo sliku koju je potrebno obraditi kako bi iz obrasca mogli izdvojiti nama zanimljive podatke. Prilikom tog postupka susrećemo se sa sljedećim problemima.

- Suvišna količina informacija.

Za reprezentaciju formulara potrebno je poznavati pozicije slikovnih elemenata zato što one definiraju oblik tablica formulara, dodatnih oznaka te znamenaka bodova. Prilikom skeniranja u slici se za svaki slikovni element spremaju tri komponente boje, crvena, zelena i plava te stupanj prozirnosti (engl. *alpha channel, transparency*). U korištenom sustavu svaka boja u memoriji i stupanj prozirnosti zauzimaju po 8 bita. Komponente boja su suvišne za segmentaciju brojeva i klasifikaciju znamenaka u postupcima opisanim u kasnijim poglavljima. Smanjenjem količine informacija za svaki slikovni element smanjujemo potrebu za memorijom prilikom obrade podataka i vrijeme potrebno za dohvat i analizu podataka prilikom daljnje obrade skenirane slike. Postupci smanjena količine suvišnih informacija opisani su u poglavlju 2.1.1 i 2.1.2.

- Nepoznavanje pozicije polja u kojima se nalaze bodovi studenata.

Nepoznata pozicija je uzrokovana razlikom u rezoluciji skeniranog obrasca od originalnog obrasca te razlikom u orijentaciji.

### 2.1.1. Pretvaranje boja u nijanse sive

Pretvaranje boja u nijanse sive je postupak kojim odbacujemo informacije o bojama pojedinog slikovnog elementa slike, no čuvamo informaciju o intenzitetu osvjetljenja. U nastavku je opisano nekoliko postupaka pretvaranja slike u boji u sliku s nijansama sive koji će biti uspoređeni u sklopu ovog rada.

- Metoda prosjeka intenziteta pojedinih boja. Iznos sive nijanse za pojedini slikovni element na slici se može odrediti sljedećim izrazom

$$s = \frac{R+G+B}{3},$$

gdje R, G, B predstavljaju redom iznose crvene, zelene i plave boje slikovnog elementa [3].

- Kolorimetrijska metoda. Metoda proširuje metodu prosjeka tako da održi luminaciju slikovnog elementa kako je vidi čovjek. Metoda radi težinski prosjek boja pri čemu uzima u obzir najveću izraženost zelene boje na koju je ljudsko oko najosjetljivije, zatim crvene te naposljetku plave boje. Težine su određene eksperimentalno te je u literaturi moguće pronaći više različitih vrijednosti, a jedna od najraširenijih je definirana dokumentom CIE1931 koji je izradila Internacionalna komisija za iluminaciju.

Iznos sive nijanse dan sljedećim izrazom preuzet je iz [3].

$$s = 0.299R + 0.587G + 0.114B$$

- Jednokanalna metoda. Metoda se zasniva na odabiru kanala boje koji će se zadržati na čitavoj slici. Na primjer ako odaberemo crveni kanal, za svaki slikovni element ćemo uzeti intenzitet crvene boje kao intenzitet sive nijanse.
- Metoda maksimuma intenziteta. Za svaki pojedini slikovni element se odabire intenzitet boje koji je na tom slikovnom elementu najveći.

$$s = \max(R, G, B)$$

- Metoda minimuma intenziteta. Za svaki slikovni element se odabire intenzitet boje koji je na tom slikovnom elementu najmanji.

$$s = \min(R, G, B)$$

- Metoda uklanjanja zasićenosti (engl. *desaturation*). Metoda prvo pretvara RGB prostor boja u cilindričnu reprezentaciju boja (HSL reprezentacija) u kojoj se svjetlina (engl. *lightness*) pojavljuje na  $z$  osi, nijansa boje (engl. *hue*) na kutu  $\theta$  te zasićenost (engl. *saturation*) na  $r$  osi. Crvena boja se nalazi na  $0^\circ$ , zelena na  $120^\circ$ , a plava na  $240^\circ$ . Metoda zatim smanjuje saturaciju na 0 i uzima vrijednost svjetline. Izračunom dobivamo izraz za iznos sive nijanse

$$s = \frac{\min(R, G, B) + \max(R, G, B)}{2}.$$

Znatželjnog čitatelja na detaljan izvod upućujemo na [3].

### 2.1.2. Binarizacija slike

Binarizacija slike je postupak kojim se slika pretvara u sliku čiji je svaki slikovni element opisan s informacijom da li je upaljen ili ugašen. Matematički problem možemo pisati kao traženje parametra  $T$  funkcije

$$b(x, y) = \begin{cases} 0, & v(x, y) < T \\ 1, & \text{inače} \end{cases}$$

gdje  $b$  predstavlja binariziranu vrijednost slikovnog elementa,  $v(x, y)$  vrijednost slikovnog elementa u rasponu od 0 do 255. Parametar  $T$  se u literaturi naziva prag binarizacije [4].

Metode traženja praga se mogu podijeliti na metode traženja globalnog praga za čitavu sliku i lokalne metode koje traže prag za pojedine dijelove ili točke na slici.

Metode globalne binarizacije su povoljnije u slučajevima ujednačenog osvjetljenja dok se metode lokalne (adaptivne) binarizacije prilagođavaju i slučajevima s neujednačenim osvjetljenjem, no zahtijevaju više izračuna.

U nastavku su dani opisi algoritama koji će biti uspoređeni u sklopu ovog rada.

- Globalna binarizacija fiksnim pragom. Za prag se odabire jedna vrijednost u rasponu intenziteta između 0 i 255. Prag se odabire proizvoljno. Kako metoda nije prilagodljiva njena uspješnost će biti sužena na užu skup slika koje su imale slično osvjetljenje i u kontekstu skeniranja obrazaca sličnu kvalitetu tiska i traga grafitne ili kemijske olovke kakvo je bilo prilikom postavljanja praga.
- Globalna binarizacija algoritmom Otsu. Metoda se zasniva na pretpostavci da na slici imamo dvije klase slikovnih elemenata. Prva klasa sadrži slikovne elemente koji predstavljaju pozadinu slike, a druga klasa sadrži slikovne elemente koji su nacrtani na toj pozadini. Metoda zatim želi maksimizirati varijancu između dviju klasa odnosno minimizirati unutar klasnu varijancu.

Ponderirana varijanca između dvije klase dana je sljedećim izrazom

$$\sigma_w^2 = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t)$$

Pri čemu su  $\sigma_1^2$  i  $\sigma_2^2$  varijance pojedinih klasa, a  $q_1$  i  $q_2$  vjerojatnosti pojavljivanja slikovnih elemenata određenog intenziteta iz pojedine klase dane sljedećim izrazima

$$q_1(t) = \sum_{i=1}^t p(i)$$

$$q_2(t) = \sum_{i=t+1}^I p(i)$$

gdje je  $t$  trenutni prag koji dijeli dvije klase slikovnih elemenata, a  $I$  maksimalni intenzitet.

Varijancu pojedinih klasa možemo po definiciji računati kao sumu kvadrata odstupanja od očekivane vrijednosti, odnosno možemo ih računati sljedećim izrazima.

$$\sigma_1^2 = \sum_{i=1}^t (i - \mu_1(t))^2 \frac{p(i)}{q_1(t)}$$

$$\sigma_2^2 = \sum_{i=t+1}^I (i - \mu_2(t))^2 \frac{p(i)}{q_2(t)}$$

Očekivanja pojedinih klasa označena su oznakama  $\mu_1$  i  $\mu_2$  te se mogu izračunati koristeći sljedeće jednakosti.

$$\mu_1(t) = \sum_{i=1}^t \frac{ip(i)}{q_1(t)}$$

$$\mu_2 = \sum_{i=t+1}^I \frac{ip(i)}{q_2(t)}$$

Ukupna varijanca nad svim slikovnim elementima je konstantna i možemo je računati kao zbroj međuklasne i unutar-klasne varijance.

$$\sigma^2 = \sigma_w^2 + \sigma_{unutar}^2$$

$$\sigma^2 = \sigma_w^2 + q_1(t)(1 - q_1(t))(\mu_1(t) - \mu_2(t))^2$$

Konačno traženi prag  $t$  dobivamo maksimiziranjem izraza međuklasne varijance  $\sigma_w^2$ , odnosno minimiziranjem unutar-klasne varijance  $\sigma_{unutar}^2$ .

Zainteresiranog čitatelja upućujemo na detaljnije razmatranje navedenog algoritma koje je dano u [5].

- Lokalna binarizacija primjenom algoritama pomičnog prozora.

Razmatrat ćemo dva srodna algoritma lokalne binarizacije tekstualnih dokumenata, algoritam Niblack i Sauvola nadogradnju algoritma Niblack.

Prilikom binarizacije ćemo koristiti prozor dimenzije  $N \times N$  unutar kojeg ćemo računati prag binarizacije  $T(x, y)$  za točku u središtu prozora. Prozor ćemo pomicati po slici te za svaki slikovni element ćemo odrediti binariziranu vrijednost. Pri takvom postupku izračuna praga za pojedini slikovni element dolazimo do problema izračuna

vrijednosti za rubne slikovne elemente slike jer prozori za izračun njihovog praga izlaze izvan slike.

Navedeni problem ćemo riješiti tako što ćemo ograničiti binarizaciju slike na slikovne elemente za koje je prozor unutar slike, a granične slikovne elemente ćemo odbaciti iz rezultatne slike. U slučaju skeniranja obrazaca s bodovima studenata taj problem neće utjecati na naš rezultat zbog dovoljno velike margine na obrascu.

Nakon što smo odredili prozor oko jednog slikovnog elementa vrijednost praga određujemo na temelju statističke obrade slikovnih elemenata unutar prozora.

Postupak Niblack [4] definira prag binarizacije sljedećim izrazom

$$T_N(x, y) = m(x, y) + k_N s(x, y),$$

pri čemu je  $m(x, y)$  prosječni intenzitet slikovnog elementa unutar prozora,  $s(x, y)$  standardna devijacija vrijednosti intenziteta, a  $k_N$  konstanta koja poprima negativne vrijednosti te je u literaturi [4] savjetovano korištenje vrijednosti  $k = -0.2$ .

Postupak Sauvola za binarizaciju tekstualnih dokumenata [11] modificira Niblackov pristup koristeći sljedeći izraz za određivanje praga binarizacije unutar prozora

$$T_S(x, y) = m(x, y) \left[ 1 + k_s \left( \frac{s(x, y)}{R} - 1 \right) \right],$$

pri čemu su prosječna vrijednost  $m(x, y)$  i standardna devijacija  $s(x, y)$  preuzeti iz Niblackovog postupka, a konstanta  $k_S$  za razliku od Niblackovog postupka poprima pozitivne vrijednosti, a  $R$  je konstanta koja se u literaturi može pronaći pod nazivom dinamički raspon standardne devijacije (engl. *dynamic range of standard deviation*). Autori algoritma su prilikom demonstracije postupka u svojem radu [11] predložili sljedeće vrijednosti parametara prilikom rada sa slikama čiji je intenzitet zapisan u 8-bitnoj preciznosti:  $R = 128, k = 0.5$ .

Postupak Sauvola pokazuje bolje ponašanje od Niblackovog postupka [12] u slučajevima gdje slikovni elementi koji pripadaju tekstu imaju vrijednosti intenziteta sive boje blizu 0, a vrijednosti pozadine blizu vrijednosti 255. U slučajevima gdje su vrijednosti intenziteta sive boje teksta blizu intenziteta pozadine rezultati degradiraju.

### 2.1.3. Određivanje orijentacije i veličine obrasca

Cilj ovog potpoglavlja je opisati postupak određivanja pozicije ćelija tablice u kojima se nalaze bodovi studenata. Pozicije polja tablice određujemo na skeniranim obrascima na temelju predloška obrasca.

Pozicije su nam nepoznate zato što prilikom skeniranja obrazac može biti zarotiran te može biti skeniran različitom rezolucijom.

Orijentaciju možemo odrediti na temelju značajki obrasca. Prvi pristup je detekcijom linija, od kojih je sastavljena tablica, detektirati zakrenutost obrasca te ga nakon toga zarotirati i obrezati crne rubove. Linije možemo detektirati pomoću Hughove transformacije. Kako je takav pristup računski skup i kako smo u obrazac ugradili dodatne značajke, crne markere, radi lakšeg određivanja orijentacije formulara ovaj pristup nećemo dalje razmatrati u okviru ovog rada.

Drugi pristup je pomoću kvadratnih crnih markera koje smo ugradili u obrazac. Prvi korak će biti algoritmom pretrage pronaći markere. Konkretni algoritam pretrage slike za značajke korišten u okviru ovoga rada dan je u poglavlju 3.1.

Definirajmo koordinatni sustav predloška obrasca. Marker u gornjem lijevom kutu označit ćemo s  $O(x_o, y_o)$ , marker u donjem lijevom kutu označit ćemo s  $Y(x_Y, y_Y)$ , a marker u gornjem desnom kutu s  $X(x_X, y_X)$ . Pomoću tri markera definiramo koordinatni sustav čije će ishodište biti u točki  $O$ , a jedinične vektore koordinatnih osi definirat ćemo sljedećim izrazima:  $\vec{i} = \frac{\overrightarrow{OX}}{|\overrightarrow{OX}|}$ ,  $\vec{j} = \frac{\overrightarrow{OY}}{|\overrightarrow{OY}|}$ .

U nastavku ćemo koristiti oznake  $\vec{u}$  i  $\vec{v}$  za jedinične vektore koordinatnog sustava papira. Definirajmo koordinatni sustav skeniranog obrasca. Marker u gornjem lijevom kutu označit ćemo s  $O_1(x_{o1}, y_{o1})$ , marker u donjem lijevom kutu označit ćemo s  $Y_1(x_{Y1}, y_{Y1})$ , a marker u gornjem desnom kutu s  $X_1(x_{X1}, y_{X1})$ . Ishodište koordinatnog sustava bit će u točki  $O$ , a jedinične vektore koordinatnih osi definirat ćemo analogno koordinatnom sustavu predloška:  $\vec{i}_1 = \frac{\overrightarrow{OX_1}}{|\overrightarrow{OX_1}|}$ ,  $\vec{j}_1 = \frac{\overrightarrow{OY_1}}{|\overrightarrow{OY_1}|}$ .

Koordinatni sustav skeniranog obrasca ćemo transformirati u koordinatni sustav predloška obrasca. Nakon toga ćemo moći iskoristiti informaciju o pozicijama ćelija za unos bodova studenata na predlošku obrasca kako bi izdvojili ćelije sa skeniranog obrasca.

Transformacije i pripadne matrice koje je potrebno primijeniti na skeniranom obrascu kako bi se transformirao koordinatni sustav dane su slijedno u nastavku [16]. Prilikom primjene transformacija na točke koristit ćemo prikaz točke u homogenom prostoru kako bi pojednostavili izračun.

- Translacija ishodišta koordinatnog sustava obrasca  $O_1$  u ishodište papira.

$$T_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x_{o1} & -y_{o1} & 1 \end{bmatrix}$$

- Rotacija jediničnih vektora koordinatnog sustava obrasca  $\vec{i}_1$  i  $\vec{i}_2$  tako da se poklope s jediničnim vektorima predloška  $\vec{i}, \vec{j}$ . Kut između dva jedinična vektora  $\vec{i}_1$  i  $\vec{i}$  možemo izračunati koristeći sljedeći izraz.

$$\cos(\alpha) = \vec{i}_1 \cdot \vec{i}$$

A smjer kuta  $\alpha$  za koji rotiramo ćemo odrediti preko odnosa  $y$  koordinata točaka  $X$  i  $X_1$ . Radi jednostavnijeg prikaza transformacije definirat ćemo funkciju  $g(x, y)$  koja će odrediti smjer rotacije. Uzeto je obzir da pri obradi slike  $y$  os gleda prema dolje.

$$g(y_X, y_{X1}) = \begin{cases} 1, & y_X < y_{X1} \\ -1, & y_X \geq y_{X1} \end{cases}$$

Tražena transformacija rotacije je dana u nastavku.

$$T_2 = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha g(y_X, y_{X1})) & 0 \\ \sin(\alpha g(y_X, y_{X1})) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Skaliranje jediničnih vektora koordinatnog sustava obrasca. Kako bi skraćeno prikazali matricu skaliranja definirat ćemo oznake  $s_x = \frac{|OX|}{|OX_1|}$ ,  $s_y = \frac{|OY|}{|OY_1|}$ .

Transformacija skaliranja dana je u nastavku.

$$T_3 = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Translacija u ishodište koordinatnog sustava obrasca  $O$ .

$$T_4 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x_o & y_o & 1 \end{bmatrix}$$

Konačnu transformaciju dobivamo slijednim umnoškom navedenih transformacija te konačnu transformaciju primjenjujemo na točke skeniranog obrasca.

$$T = T_1 T_2 T_3 T_4$$



Nakon transformiranja skeniranog obrasca izdvajamo ćelije koje sadrže bodove studenata koristeći koordinate ćelija iz predloška obrasca. Izdvojene ćelije zatim prosljeđujemo sustavu za segmentaciju brojeva.

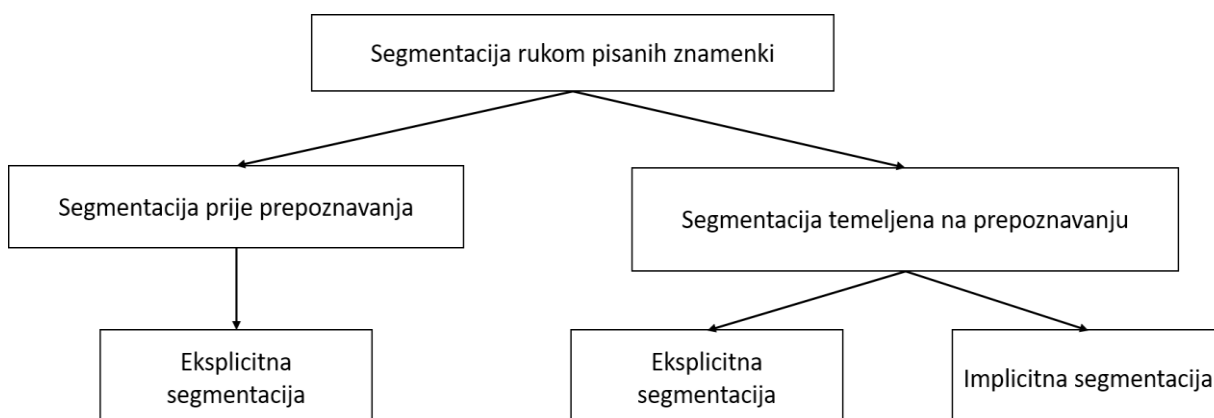
Postupak možemo i prilagoditi tako da primijenimo  $T^{-1}$  na točke koje opisuju ćelije osnovnog obrasca te izravno izrezati ćelije iz skeniranog obrasca bez transformiranja slike.

## 2.2. Segmentacija znamenki

Problem prepoznavanja broja kod područja s nepoznatim brojem znamenki koje nisu podijeljene u predefinirane regije kakve možemo naći na starijim formularima gdje je svaki broj bio pisan u svoju kućicu težak je problem [15]. Takav problem želimo podijeliti na potprobleme prepoznavanja pojedinih znamenki te zato početni broj trebamo segmentirati. Problem koji susrećemo u okviru ovog rada se u literaturi može pronaći pod nazivom segmentiranje neodređeno dugog niza rukom pisanih znamenki (engl. *segmentation of unconstrained handwritten numeral strings*).

Pristupi segmentaciji mogu biti podijeljeni u tri razreda: eksplicitna segmentacija temeljena na klasifikaciji (engl. *recognition-based segmentation*), implicitna (holistička) segmentacija temeljena na klasifikaciji (engl. *implicit segmentation, holistic segmentation methods*) [6].

Grafički prikaz pristupa segmentaciji brojeva dan je na slici 3. [7].



Slika 3. Pristupi segmentaciji (prilagođeno iz [7])

Segmentacija temeljena na klasifikaciji stvara nekoliko potencijalnih hipoteza segmentacije koje zatim prosljeđuje klasifikatoru kako bi mogao odrediti koja je hipoteza najbolja. Korišteni klasifikator može biti klasifikator znamenki ili neki posebno prilagođeni klasifikator kao što je klasifikator temeljen na prepoznavanju klasa dodirivanja znamenki [7]. Implicitna segmentacija je pristup u kojem se segmentacija implicitno odvija prilikom procesa klasifikacije. Klasifikatoru se prilikom implicitne segmentacije predaje čitav broj. Pristupi temeljeni na neovisnoj segmentaciji stvaraju jednu hipotezu i nisu ovisni o klasifikatoru znamenki.

U okviru ovog rada bit će predstavljeni dva algoritma eksplicitne segmentacije neovisne o klasifikatoru.

- Segmentacija temeljena na vertikalnom histogramu.

Algoritam prvo računa vertikalnu distribuciju slikovnih elemenata (histogram) binarizirane slike broja. Vertikalna distribucija slikovnih elemenata binarne slike koja ima širinu  $w$  i visinu  $h$  računa se na sljedeći način. Za svaki stupac slike broji se koliko ima crnih slikovnih elemenata u tom stupcu te se taj broj podjeli s brojem slikovnih elemenata u jednom stupcu slike.

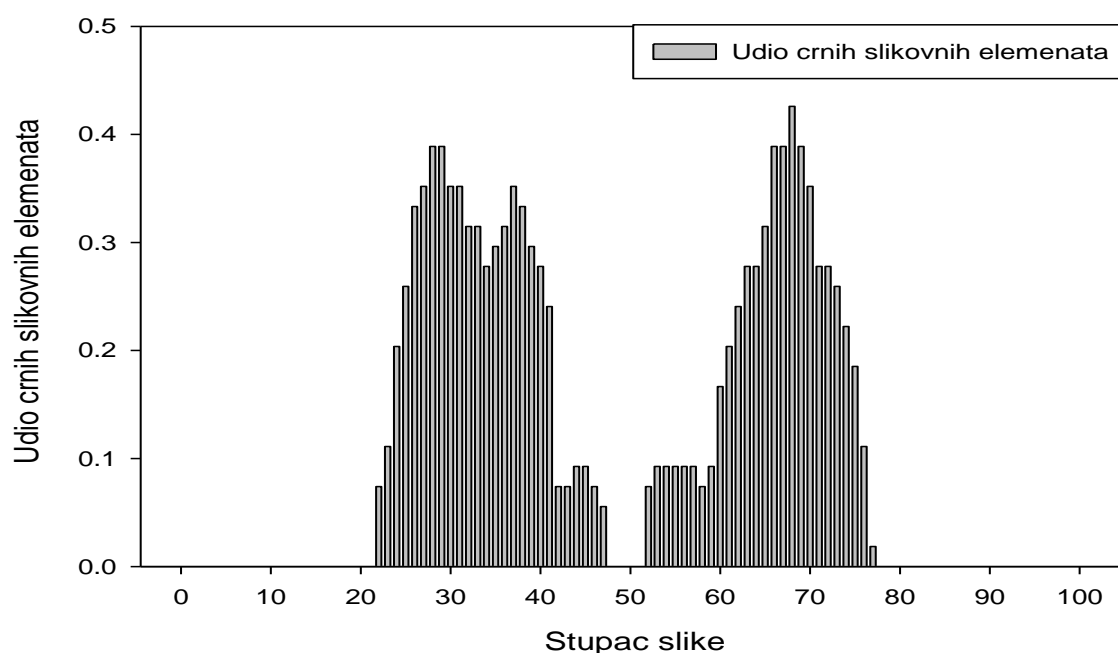
Matematički vertikalnu projekciju možemo definirati na sljedeći način. Sliku ćemo predstaviti kao diskretnu funkciju  $B(i, j)$  uz  $0 \leq i < w, 0 \leq j < h$ . Vertikalnu projekciju  $H(x)$  zatim računamo tako da fiksiramo  $i$  na neku vrijednost  $x$  horizontalne osi slike te izvodimo sljedeći izraz [8]

$$H(x) = \sum_{j=0}^{h-1} B(x, j).$$

Znamenke ćemo segmentirati tako da ćemo izdvojiti područja gdje je vertikalni histogram različit od 0. Primjeri izračunatih distribucija slika binarnih brojeva danih na slikama 4, 5 i 6 prikazani su grafovima 1, 2 i 3 pomoću vertikalnog histograma.

53

Slika 4. Primjer slike broja s odvojenim znamenkama

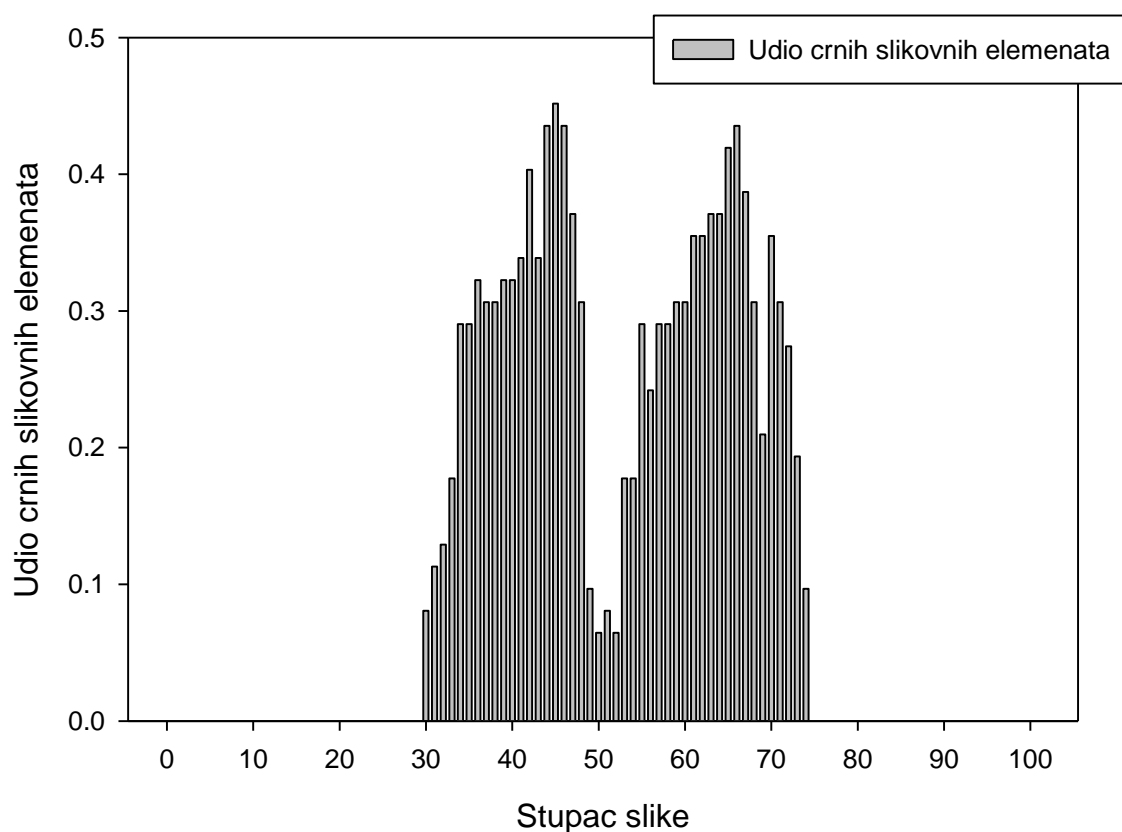


Graf 1. Vertikalni histogram binarizirane slike 4.

Slika 4 prikazuje dvije rukom pisane znamenke koje su međusobno odvojene i čije se vertikalne projekcije ne preklapaju. Na grafu 1 možemo uočiti kako se znamenka „5“ nalazi na području od 22. do 47. stupca slike, dok se znamenka „3“ nalazi na području od 52. do 77. stupca slike. Na ovom primjeru vidimo kako je pomoću vertikalne projekcije moguće segmentirati brojeve koji se ne dodiruju te čije se vertikalne projekcije ne preklapaju.

53

Slika 5. Primjer slike broja s prekrivenim znamenkama

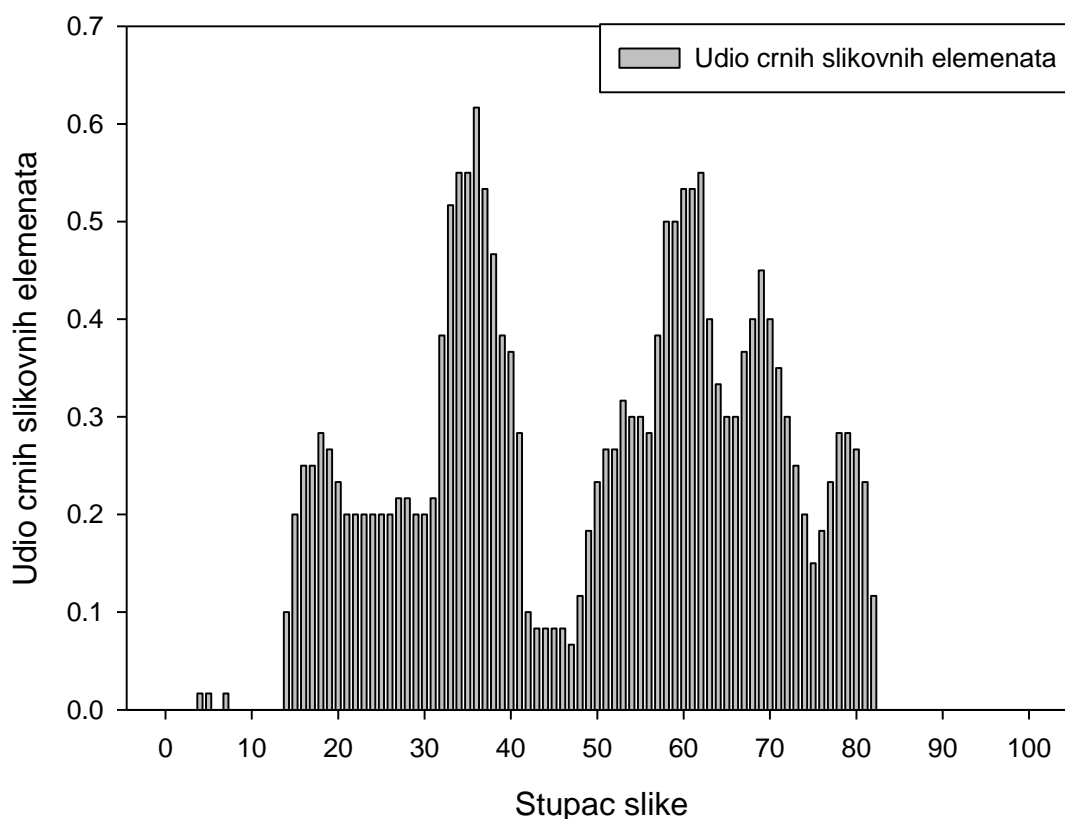


Graf 2 Vertikalni histogram binarizirane slike 5.

Slika 5 prikazuje dvije rukom pisane znamenke „53“. Znamenka „5“ prekriva znamenku „3“ na području od 54. do 65. stupca slike. Na grafu 2 možemo vidjeti histogram slike 5. Na temelju histograma vidimo kako ne možemo odvojiti znamenke koje se međusobno preklapaju.



Slika 6. Primjer slike broja sa spojenim znamenkama



Graf 3. Vertikalni histogram slike 6.

Slika 6 prikazuje dvije znamenke „78“ koje su međusobno spojene. Takve znamenke ne možemo segmentirati na temelju histograma prikazanog na grafu 3. Ujedno na grafu 3 možemo vidjeti kako se na početku slike između 4. i 7. stupca nalaze nekoliko stupaca koji predstavljaju suvišne slikovne elemente koje bi trebali ignorirati tako da stavimo granicu minimalnog udjela slikovnih elemenata u vertikalnoj projekciji koji mogu činiti znamenku. Pri odabiru granice moramo paziti kako ne bi odbacili decimalnu točku ili decimalni zarez.

- Segmentacija temeljena na povezanim komponentama.

Algoritam se temelji na grupiranju slikovnih elemenata u međusobno povezane objekte koji se nalaze na bijeloj podlozi. Grupiranje se odvija tako da se svakom slikovnom elementu

dodjeljuje oznaka koja pripada povezanoj komponenti unutar koje je slikovni element [9][10].

Komponente unutar regija koje ćemo analizirati će predstavljati znamenke broja. Primjeri segmentiranja brojeva nad slikama 4, 5 i 6 koristeći algoritam označavanja povezanih komponenti dani su na sljedećim slikama.



Slika 7. Primjer segmentacije slike 4. algoritmom povezanih komponenti

Osnovni slučaj segmentiranja broja sastavljenog od dvije znamenke prikazan je na slici 7. Algoritam je označio prvu povezanu komponentu sivom bojom, a ta komponenta predstavlja broj „5“. Druga komponenta je označena crvenom bojom te predstavlja broj „3“.



Slika 8. Primjer segmentacije slike 5. algoritmom povezanih komponenti

Slika 8. prikazuje segmentaciju znamenki koje se preklapaju, no nisu povezane. Algoritam uspješno odvaja broj „5“ koji je označen sivom bojom i broj „3“ koji je označen plavom bojom u različite komponente, no algoritam je prepoznao tri komponente. Treća komponenta je označena crvenom bojom i predstavlja nekoliko suvišnih slikovnih elemenata na slici, odnosno šum. Kako bi riješili problem suvišnih malih nakupina slikovnih elemenata koje nisu znamenke niti decimalni separatori u okviru ovog rada ćemo predložiti način filtriranja potencijalnih kandidata komponenti.



Slika 9. Primjer segmentacije slike 6. algoritmom povezanih komponenti

U slučaju povezanih znakova koristeći ovaj algoritam nećemo moći razdvojiti znamenke jer će one biti označene kao jedna komponenta. Navedeni slučaj se može vidjeti na slici 9 gdje su i znamenke „7“ i „8“ označene plavom bojom.

Algoritam korišten u okviru ovog rada temeljen je na dvoprolaznom algoritmu označavanja komponenti preuzetog iz [10] te je objašnjen u poglavlju 4.

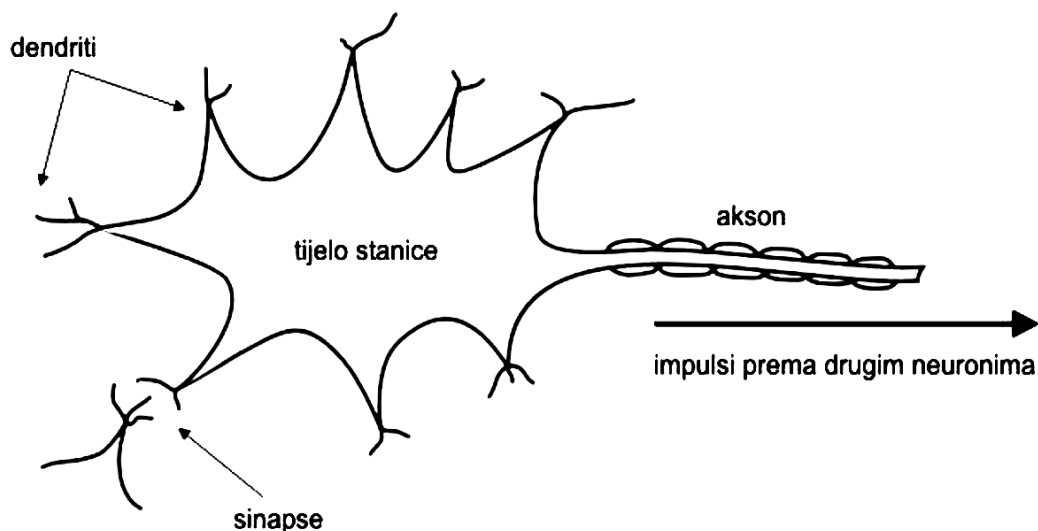
## 2.3. Klasifikacija znamenki

Nakon segmentacije znakova, svaku znamenku i znakove odvajanja znamenki (decimalni zarez ili točku) je potrebno klasificirati. Klasifikaciju je moguće raditi na više načina, a u sklopu ovog rada odabrana je metoda klasifikacije pomoću unaprijedne slojevite neuronske mreže. U nastavku ovog poglavlja dan je opis neuronske mreže i učenje gradijentnim spustom algoritmom propagacije pogreške unatrag (engl. *Backpropagation*).

### 2.3.1. Model umjetne neuronske mreže

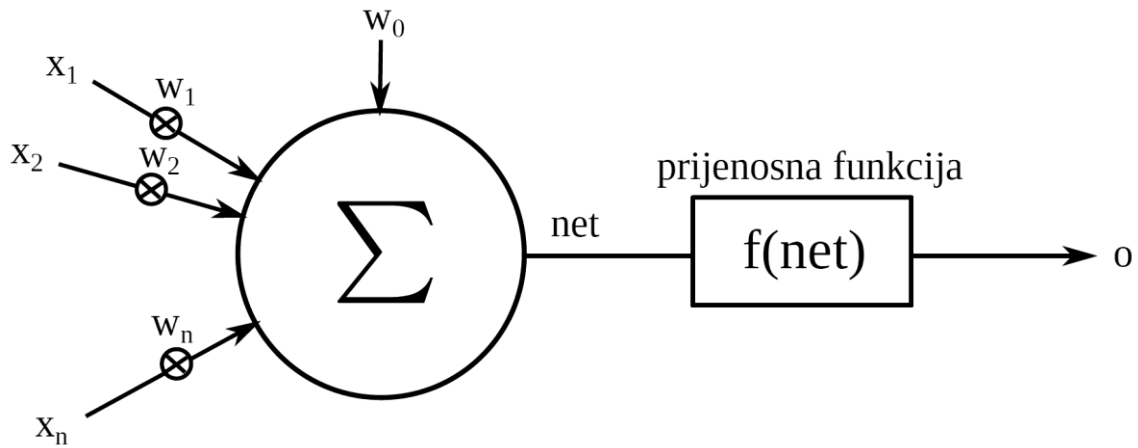
Model umjetne neuronske mreže je model nastao željom za oponašanjem rada ljudskog živčanog sustava. Neuronske mreže su korištene u mnogim primjenama od kojih su neke obrada jezika, prepoznavanje pisanih znakova, prepoznavanje uzoraka i druge.

Neuronska mreža je građena od povezanih neurona. Neuron je modeliran nalik na biološki neuron. Pojednostavljeno biološki neuron se sastoji od dendrita, tijela neurona i aksona. Prikaz biološkog neurona dan je na slici 10. Dendriti predstavljaju „ulaz“ u neuron na kojem se sakuplja naboj. Nakon što se akumulira dovoljna količina naboja dolazi do proboja u kojem naboj prolazi kroz tijelo neurona te u konačnici izlazi na mjestu aksona koji predstavlja „izlaz“ iz neurona [13].



Slika 10. Biološki neuron (prilagođeno iz [14])

Model umjetnog neurona dan je slikom 11. Ulazi  $\vec{x} = (1, x_1, x_2, \dots, x_n)$  su analogni ulazima u dendrite. Prijenosna funkcija  $f(net)$  određuje prag okidanja neurona te je analogna tijelu neurona. Težine  $\vec{w} = (w_0, w_1, \dots, w_n)$  određuju u kojoj mjeri svaki od ulaza pobuđuje neuron. Izlaz  $y$  je analogan izlaznom aksonu iz neurona.



Slika 11. Model umjetnog neurona (preuzeto iz [14])

Ukupna pobuda neurona dana je sljedećim izrazom.

$$net = \vec{w} \cdot \vec{x}$$

Neurone možemo podijeliti prema prijenosnoj funkciji od koji su tipični primjeri dani u nastavku.

- TLU-perceptron (engl. *Threshold Logic Unit*). Ovaj oblik neurona su koristili W. McCulloch i W. Pitts 1943. godine, te su koristeći njega pokazali kako je koristeći TLU-perceptron moguće izračunati osnovne Booleove funkcije I, ILI i NE. Prijenosna funkcija koju koristi TLU-perceptron je funkcija skoka. Funkcija skoka je definirana izrazom:

$$f(net) = \begin{cases} 0, & net < 0 \\ 1, & net \geq 0 \end{cases}$$

- Neuron s linearnom prijenosnom funkcijom. Prijenosna funkcija je oblika

$$f(net) = k \cdot net + l$$

gdje su  $k$  i  $l$  konstante.

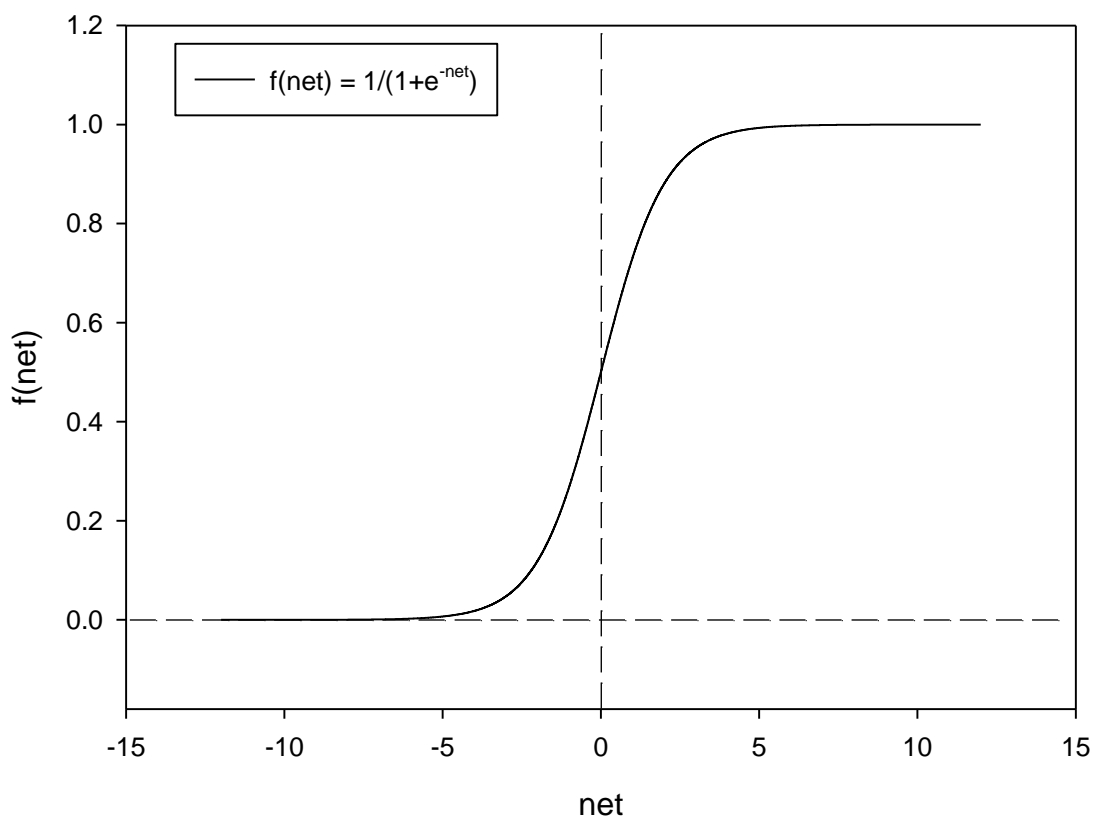
Poseban slučaj neurona s linearnom prijenosnom funkcijom je jedinični neuron koji preslikava ulaz na izlaz, odnosno ima konstante  $k = 1$  i  $l = 0$ . Jedinične neurone koristimo kao ulazne neurone u neuronsku mrežu.



- Neuron sa sigmoidalnom (logističkom) funkcijom. Prijenosna funkcija je definirana sljedećim izrazom.

$$f(net) = \frac{1}{1 + e^{-net}}$$

Neuroni sa sigmoidalnom prijenosnom funkcijom su nam korisni jer je prijenosna funkcija derivabilna te se unaprijedne neuronske mreže sastavljene od takvih neurona mogu trenirati algoritmom propagacije pogreške unatrag. Graf sigmoidalne funkcije dan je na grafu 4.



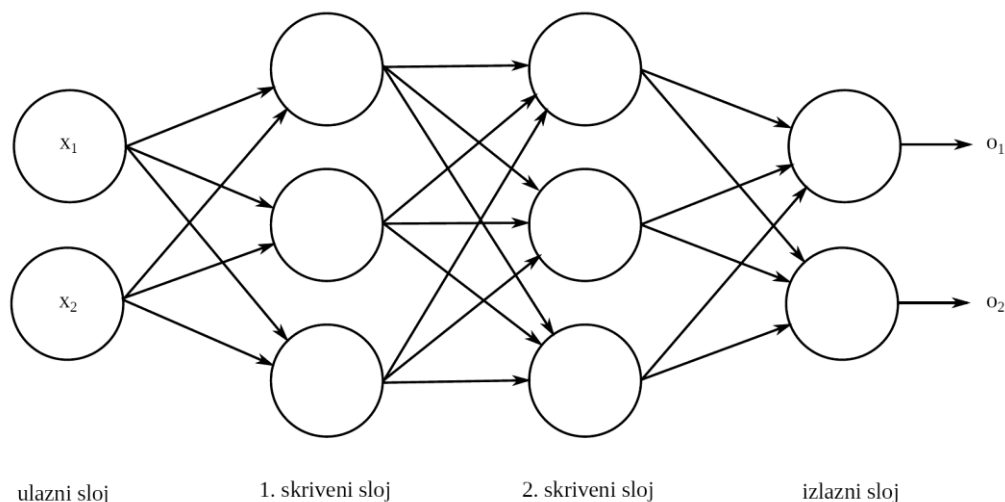
Graf 4. Prikaz sigmoidalne funkcije

Derivacija sigmoidalne funkcije je dana sljedećim izrazom.

$$\frac{df(net)}{net} = f(net) \cdot (1 - f(net)) \quad (1)$$

U ovom radu bit će korištena unaprijedna slojevita neuronska mreža. Primjer unaprijedne slojevite neuronske mreže dan je na slici 12. Neuronska mreža je unaprijedna ako i samo ako nema ciklusa. Slojevita neuronska mreža je neuronska mreža u kojoj za svaki sloj  $k$  vrijedi da se izlazi iz tog sloja računaju samo i isključivo

na temelju izlaza neurona sloja  $k - 1$ . Veze između neurona u istom sloju nisu dopuštene.



Slika 12. Slojevitá neuronska mreža arhitekture  $2 \times 3 \times 3 \times 2$  (preuzeto iz [14])

Prvi sloj neuronske mreže se naziva ulazni sloj te se on sastoji od onoliko neurona koliko imamo ulaza. Prijelazna funkcija ulaznih neurona je funkcija identiteta  $f(net) = net$ .

Svi slojevi između prvog i zadnjeg sloja nazivaju se skrivenim slojevima.

Zadnji sloj je izlazni sloj te je broj neurona u zadnjem sloju jednak broju izlaza koje očekujemo od neuronske mreže.

### 2.3.2. Učenje umjetne neuronske mreže

Rad s umjetnom neuronskom mrežom možemo podijeliti u dvije faze: fazu učenja i fazu iskorištavanja (obrade podataka). Učenje neuronske mreže se temelji na spoznajama koje je britanski biolog Donald Hebb predstavio 1949. u svojoj knjizi „The Organization of Behaviour“. Ideja D. Hebba je da učiti znači mijenjati jakosti veza između neurona. Tu ideju je 1958. Frank Rosenblatt spojio s McCulloch-Pitts modelom TLU-perceptrona i definirao algoritam učenja TLU-perceptrona [13].

Postupak učenja možemo promatrati kao iterativni postupak predočavanja ulaznih primjera pri čemu dolazi do postupnog prilagođavanja težina veza neurona. Sadržaj ulaznog primjera za učenje možemo podijeliti prema dva tipa učenja [14].

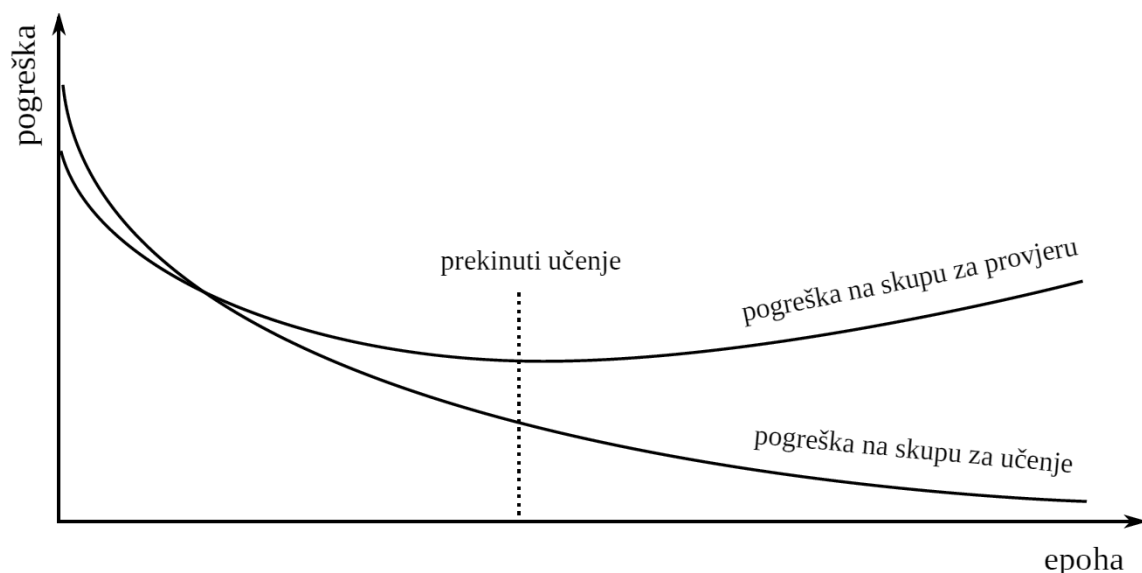
- Učenje s učiteljem (engl. *supervised learning*). Kod učenja s učiteljem ulazni primjeri su oblika (*ulaz, očekivani izlaz*).
- Učenje bez učitelja (engl. *unsupervised learning*). Kod učenja bez učitelja izlaz je *a priori* nepoznat te su ulazni primjeri oblika (*ulaz*).

U okviru ovog završnog rada razmatrat ćemo postupak učenja s učiteljem (nadziranog učenja).

Kako bismo mogli odrediti stupanj naučenost neuronske mreže skup primjera ćemo podijeliti u tri podskupa: skup za učenje (engl. *training set*), skup za provjeru (engl. *validation set*) i skup za testiranje (engl. *testing set*). Literatura [14] savjetuje da skup za učenje čini 60% početnog skupa, skup za provjeru 20%, te skup za testiranje preostalih 20% uzoraka.

Uzorke iz skupa za učenje ćemo predočavati neuronskoj mreži te ćemo na temelju tih uzoraka mijenjati težine veza između neurona. Skup za provjeru ćemo koristiti kako bi detektirali trenutak kada mreža počne gubiti sposobnost generalizacije i kada se počne prilagođavati posebnostima skupa za učenje. Skup za testiranje ćemo koristiti nakon što završimo učenje neuronske mreže kako bi odredili kvalitetu rada mreže.

Jedno predočavanje svih uzoraka podskupa naziva se epohom. Određivanje epohe u kojoj ćemo prekinuti učenje prikazano je na slici 13 koja prikazuje kretanje pogreške pri učenju neuronske mreže na skupu za učenje i skupu za provjeru.



Slika 13. Određivanje granice učenja (preuzeto iz [14])

### 2.3.3. Učenje algoritmom propagacije pogreške unatrag

Algoritam propagacije pogreške unatrag najpoznatiji je algoritam za učenje unaprijednih neuronskih mreža koji se temelji na algoritmu traženja minimuma funkcije gradijentnim spustom [14].

U okviru ovog završnog rada razmatranje algoritma propagacije pogreške unatrag ograničit ćemo na učenje unaprijedne slojevite neuronske mreže koja je sastavljena od sigmoidalnih neurona. Algoritam propagacije pogreške unatrag općenito je primjenjiv na općenitu kategoriju unaprijednih neuronskih mreža koje su sastavljene od neurona čija je prijenosna funkcija derivabilna.

Na raspolaganju imamo neuronsku mrežu koja ima  $d$  ulaza,  $m$  izlaza i  $N$  uzoraka za učenje. Uzorci za učenje su u obliku parova  $(\vec{x}_i, \vec{t}_i)$  gdje je  $\vec{x}_i$   $d$ -dimenzijski vektor koji predstavlja  $i$ -ti ulazni uzorak, a  $\vec{t}_i$   $m$ -dimenzijski vektor koji predstavlja željeni odziv mreže za  $i$ -ti ulazni uzorak [13]. Izlaze koje generira neuronska mreža za ulazni uzorak  $\vec{x}_i$  označit ćemo s  $\vec{o}_i$ .

Kriterijska funkcija  $E$  koja mjeri kvalitetu neuronske mreže, odnosno iznos pogreške dana je sljedećim izrazom

$$E = \frac{1}{2N} \sum_{s=1}^N \sum_{i=1}^m (t_{s,i} - o_{s,i})^2.$$

Konceptualno funkcija računa prosječno kvadratno odstupanje izlaza neuronske mreže od očekivane vrijednosti podijeljeno s 2. Faktor  $\frac{1}{2}$  je uzet zato što će se taj faktor poništiti prilikom derivacije kriterijske funkcije s potencijom izraza  $(t_{s,i} - o_{s,i})$  te će nam to olakšati daljnji izračun.

Cilj postupka učenja će biti korištenjem pravila gradijentnog spusta minimizirati iznos funkcije  $E$ . Primjenom pravila gradijentnog spusta dolazimo do pravila ažuriranja težina preuzetog iz [13] te prikazanog sljedećim izrazom

$$\omega_{ij}^{(k)} \leftarrow \omega_{ij}^{(k)} - \psi \frac{\partial E}{\partial \omega_{ij}^k},$$

koji definira iznos ažuriranja težine između neurona  $i$  u  $k$ -tom sloju i neurona  $j$  u  $(k + 1)$  sloju. Varijabla  $\psi$  predstavlja malu pozitivnu konstantu koja će kontrolirati veličinu korekcije težine.

Primjena pravila ažuriranja težina u algoritmu propagacije unatrag dana je u algoritmu 1. Unutar algoritma se koristi oznaka  $\eta$  koja predstavlja stopu učenja koja je definirana izrazom  $\psi \frac{1}{N}$ . Zainteresiranog čitatelja upućujemo na literaturu [13] i [14] za detaljni izvod i proučavanje algoritma.

```

Inicijaliziraj težinske faktore na slučajne vrijednosti
Dok nije ispunjen uvjet zaustavljanja čini
    Za svaki  $(\mathbf{x}, \mathbf{t})$  iz  $D$  čini
        Izračunaj izlaz  $o_u$  za svaku jedinicu  $u$ 
        Za svaku izlaznu jedinicu  $k$  izračunaj pogrešku  $\delta_k$ 
             $\delta_k \leftarrow o_k(1 - o_k)(t_k - o_k)$ 
        Za svaku skrivenu jedinicu izračunaj pogrešku  $\delta_h$ 
             $\delta_h \leftarrow o_h(1 - o_h) \sum_{s \in \text{Downstream}(h)} \omega_{hs} \delta_s$ 
        Ugodi svaki težinski faktor  $\omega_{ij}$ 
             $\omega_{ij} \leftarrow \omega_{ij} + \Delta \omega_{ij}$ 
        gdje je  $\Delta \omega_{ij} = \eta \delta_j o_i$ 
    Kraj
Kraj

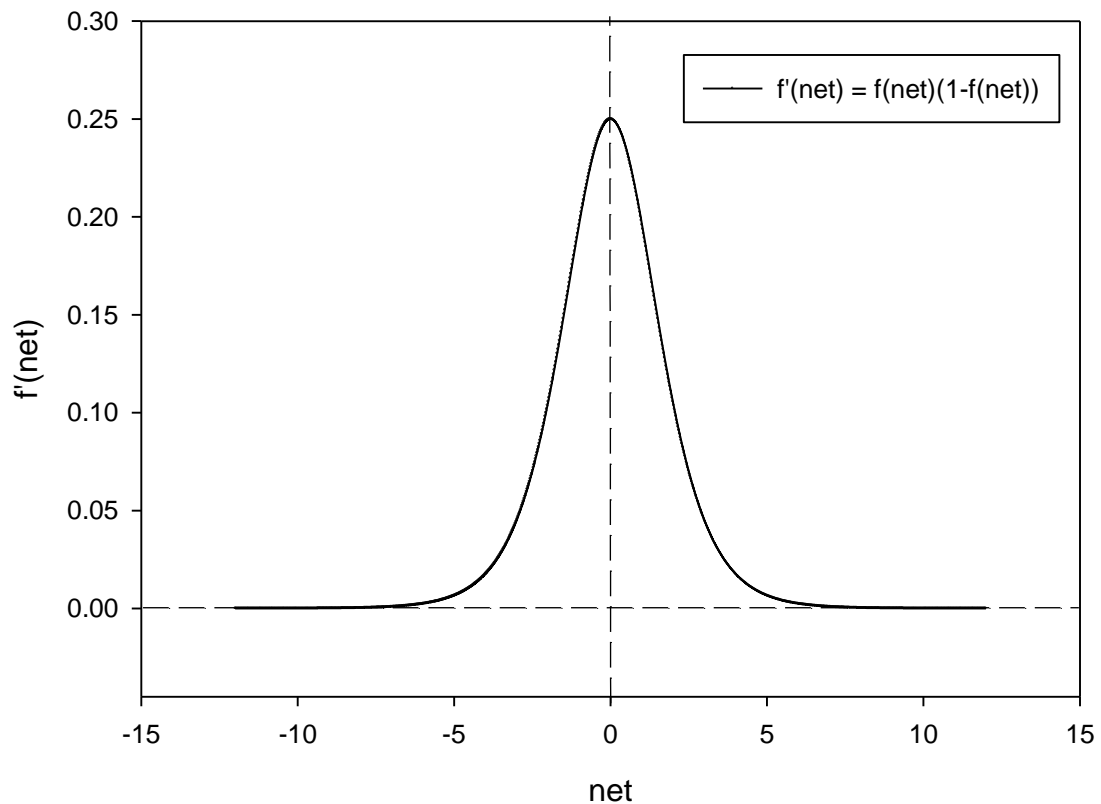
```

Algoritam 1. Algoritam propagacije pogreške unatrag (preuzet iz [14])

Kako bismo ubrzali učenje algoritmom propagacije pogreške unatrag potrebno je prilagoditi postavljanje inicijalnih težina. Početne težine se postavljaju slučajno, no granice slučajnih vrijednosti mogu značajno utjecati na brzinu kojom mreža počinje učiti. Primjer utjecaja na učenje neuronske mreže korištene u ovom radu je dan u sklopu poglavlja 5.

Pretpostavimo da imamo jedan sigmoidalni neuron s  $m$  ulaza. Njegov izlaz definiran je izrazom  $o = f(\text{net}) = \frac{1}{1+e^{-\text{net}}}$ , a ukupni ulaz  $\text{net}$  možemo pisati kao  $\sum_{i=1}^m \omega_i x_i + \omega_0$ .

Korekcija težine iz algoritma dana je izrazom  $\Delta \omega_{ij} = \eta \delta_j o_i$  koji u sebi unutar  $\delta_j$  sadrži derivaciju prijenosne funkcije. Derivaciju sigmoidalne funkcije smo definirali izrazom (1), a njen graf je prikazan na grafu 5.



Graf 5. Prikaz derivacije sigmoidalne funkcije

Na grafu možemo vidjeti kako derivacija prijenosne funkcije već i za relativno male vrijednosti *net*-a poprima vrlo mali iznos te tada znatno smanjuje korekciju težine. U literaturi [13] se predlaže da se prilagodbom težina pokuša smanjiti početni *net* na vrijednosti u rasponu između  $-2.4$  i  $2.4$ . Za  $net = 2.4$  derivacija prijenosne funkcije poprima vrijednost nešto manju od 0.8.

U sljedećim izrazima dan je izvod za ograničavanje raspona težine na ulazu neurona uz pretpostavku da je ulaz u neuron ograničen s  $+1$ . Granica na ulaznu vrijednost u iznosu od  $+1$  je u skladu s maksimalnom vrijednošću koju poprima prijenosna sigmoidalna funkcija.

$$|net| \leq 2.4$$

$$\left| \sum_{i=1}^m \omega_i x_i + \omega_0 \right| \leq 2.4$$

$$\left| \sum_{i=1}^m \omega_i + \omega_0 \right| \leq 2.4$$

$$|(m+1)\omega| \leq 2.4$$

$$|\omega| \leq \frac{2.4}{m+1}$$

Konačni rezultat nam definira raspon inicijalne težine  $\left[ -\frac{2.4}{m+1}, \frac{2.4}{m+1} \right]$ .

### 3. Priprema skeniranog obrasca

U ovom poglavlju opisan je postupak pripreme skeniranog obrasca i izdvajanje regija s bodovima studenata kako bi bodovi mogli biti segmentirani te zatim klasificirani.

Priprema obrasca obuhvaća definiranje značajki originalnog obrasca koje su navedene u poglavlju 3.1 te načine odbacivanja suvišnih informacija i određivanja orijentacije slike koji su teoretski obrađeni u poglavlju 2.1.

#### 3.1. Definiranje obrasca

Kako bi pojednostavili analizu skeniranih obrazaca želimo omogućiti korisniku unos informacija o predlošku obrasca. Predložak koji će biti korišten u okviru ovog rada prikazan je na slici 1 na stranici 2. Informacije koje tražimo od korisnika su položaji crnih oznaka koje definiraju koordinatni sustav obrasca te položaji ćelija tablice u koje se unose studentski bodovi.

Korisniku ćemo dati mogućnost da unese pozicije unutar oznaka nakon čega ćemo odrediti koordinate središta oznaka i njezine dimenzije. Način unosa u implementiranom sustavu opisan je u poglavlju 6.1. Nakon što korisnik unese osnovne informacije o oznakama, određivanje pozicije oznaka na novim skeniranim obrascima ćemo napraviti koristeći algoritam 2 koji koristi prilagođene algoritme pretraživanja dane u algoritmu 3 i algoritmu 5.

```
//podaci korišteni za inicijalizaciju detektora
poz[] ← pretpostavljene pozicije oznaka
dim ← pretpostavljena dimenzija oznake

//niz definiranih razreda koji sadrže funkciju za detekciju
detektori_oznaka ← niz referenci na inicijalizirane razrede detektora
za svaki detektor ponavlja:
    oznake = detektor.detekcijaOznaka()
    ako duljina(oznake) == 3:
        vrati oznake
vrati GREŠKA
```

Algoritam 2. Detekcija oznaka na novoskeniranim obrascima



Algoritmi pretpostavljaju da je dulja stranica vertikalna, da se oznake nalaze u rubovima stranice te da su oznake crne dok je pozadina stranice bijela.

Algoritam 3 pretražuje prošireno područje početne pozicije oznaka za crne slikovne elemente. Nakon što algoritam pronađe prvi crni slikovni element unutar regije prekida trenutnu pretragu i počevši od početnog crnog slikovnog elementa traži sve spojene crne slikovne elemente koji čine oznaku koristeći prilagođeni algoritam nastajanja područja (engl. *flood fill algorithm*) [17] koji je dan u algoritmu 4, a preinačen je tako da umjesto da mijenja boje područja sprema pozicije područja u niz.

Koristeći pozicije crnih slikovnih elemenata računamo središte oznake koristeći sljedeće izraze:

$$x_c = \frac{1}{n} \sum_{i=1}^n x_i,$$
$$y_c = \frac{1}{n} \sum_{i=1}^n y_i,$$

pri čemu  $(x_c, y_c)$  predstavljaju koordinate središta oznake,  $(x_i, y_i)$  koordinate  $i$ -te točke oznake, a  $n$  predstavlja broj točaka od kojih se oznaka sastoji.

```

funkcija detekcijaOznaka (slika, pozicije_oznaka, dim_oznaka):
    nove_oznake = lista()
    ponavljaj za sve pozicije_oznaka:
        poz ← pozicija oznake

        //ako je pretpostavljena pozicija sadržana u oznaci
        ako je slika[poz] == crna:
            nove_oznake.dodaj(centar(floodFill(poz)))
            nastavi

        //određivanje regije pretrage oznake
        poc_x = max(0, poz.x-dim_oznaka/2)
        poc_y = max(0, poz.y-dim_oznaka/2)
        kraj_x = min(slika.sirina(), poz.x+dim_oznaka/2)
        kraj_y = min(slika.visina(), poz.y+dim_oznaka/2)

        //pretraga regije
        za i=poc_x, i<kraj_x, i++ ponavljaj
            za j=poc_y, j<kraj_y, i++ ponavljaj
                ako je slika[Tocka(i,j)] == crna:
                    nove_oznake.dodaj(centar(floodFill(Tocka(i,j))))
                    nastavi iduca_pozicija_oznake

    vрати nove_oznake

struktura Tocka:
    int x
    int y

//izračun centra niza točaka
funkcija centar (Tocka[] niztocaka):
    sumx = 0
    sumy = 0
    za svaku tocku iz niztocaka:
        sumx += tocka.x
        sumy += tocka.y
    sumx = sumx/duljina(niztocaka)
    sumy = sumy/duljina(niztocaka)
    vрати Tocka(sumx, sumy)

```

Algoritam 3. Detekcija oznaka pretragom slike u definiranom području

```

ciljna_boja = crna
zamjenska_boja = bijela

//definicija susjedstva
susjed_x = {0, 1, -1, 0}
susjed_y = {1, 0, 0, -1}
susjedstvo = 4

funkcija floodFill(Tocka p1, Slika s):
    ako s[p1] == ciljna_boja:
        vрати []

    Q ← prazan red za čuvanje budućih stanja
    Q.dodaj(p1)
    t ← prazna lista točaka
    t.dodaj(p1)
    dok Q nije prazan ponavljaj:
        n ← skidamo element iz reda Q
        za i=0; i<susjedstvo; i++ ponavljaj:
            x=n.x+susjed_x[i]
            y=n.y+susjed_y[i]
            susjed = Tocka(x,y)
            ako x<0 ili y<0 ili x >= s.sirina ili y>=s.visina:
                nastavi
            ako s[susjed] == ciljna_boja:
                Q.dodaj(susjed)
                t.dodaj(susjed)

    vрати t

```

Algoritam 4. Algoritam nastajanja područja (engl. *flood fill algorithm*) (prilagođeno iz [17])

```

funkcija detekcijaOznaka (slika):
    Tocka pocetak ← dohvati prvi bijeli slikovni element
    Tocke[][] regije ← dohvati tocke regija
    //dohvat se odvija modifikacijom flood fill algoritma
    // koji pamti točke su imale bar jednog bijelog susjeda

    Tocke[][] potencijalne_oznake ← filtriraj regije
    //korišteni filter odbacuje sve regije koje nemaju barem
    //10 slikovnih elemenata visine i širine, te one koji imaju
    //visinu veću od desetine visine slike, odnosno širinu veću od
    //desetine širine slike

    ako duljina(potencijalne_oznake) <3:
        vrati GREŠKA

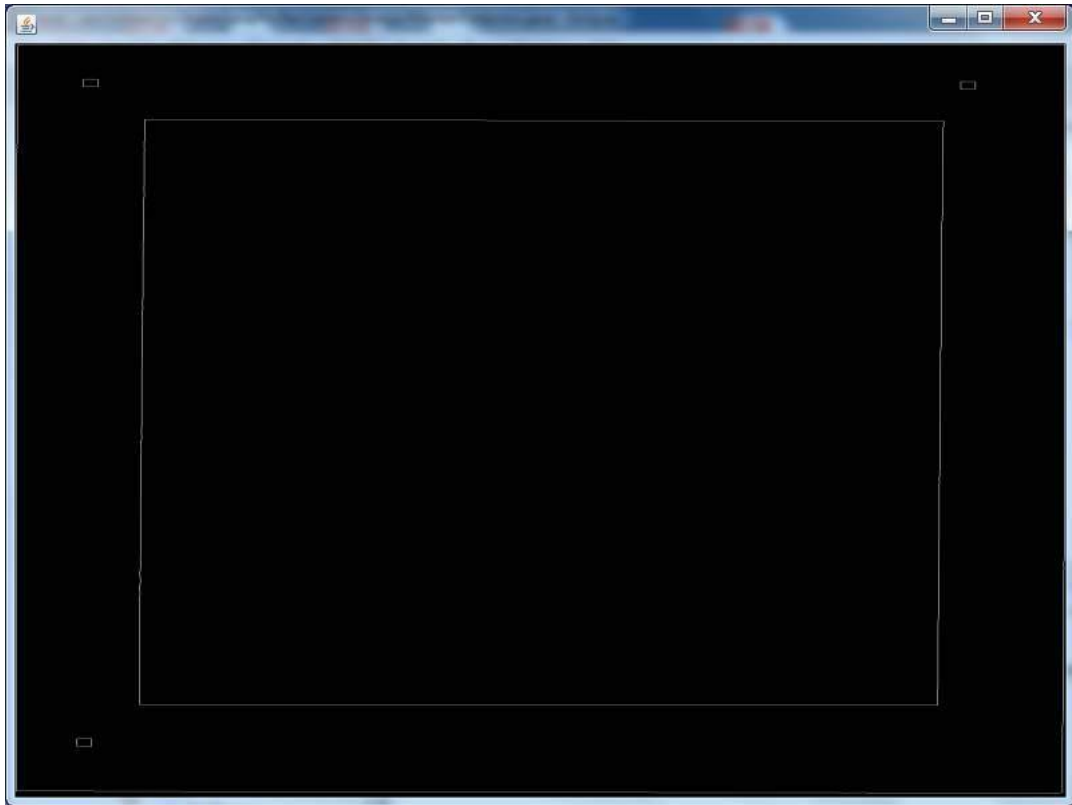
    sortiraj(potencijalne_oznake) // po površini
    Tocke[][] oznake ← nađi prve tri regije slične površine

    Tocke[] centar_oznaka
    za svaki oznaka u oznake ponavljaj:
        centar_oznaka.dodaj(centar(oznaka))
    vrati centar_oznaka

```

#### Algoritam 5. Detekcija markera pretraživanjem rubova regija

Algoritam 5 traži i povezuje rubove zadane regije. Pretraživanje regije određuju granične točke i početna točka. Granične točke su točke čiji susjedi su različite boje. Početnu točku smo definirali na temelju konkretnih ulaznih primjera gdje za početnu točku uzimamo prvu najvišu točku bijele boje. Izlaz pretraživanja su spojene granice regija, a primjer vizualizacije rezultata dan je na slici 14.



Slika 14. Granice regija dobivene algoritmom 5

Očekivane regije temeljene na predlošku obrasca danog na slici 1 su: tri regije koje sadrže oznake, regija koja sadrži tablicu te vanjske regije. Tražene regije određujemo tako da tražimo tri regije slične veličine počevši od najmanjih regija, pri čemu prvo odbacimo regije koje su premale ili prevelike da budu oznake. Dimenzije koje koristimo za odbacivanje premalih i prevelikih regija temeljimo na početnim dimenzijama oznaka i odnosu veličina slike osnovnog obrasca i skeniranog obrasca.

## 3.2. Odbacivanje suvišnih informacija

Proces odbacivanja suvišnih informacija opisan u poglavlju 2 sastoji se od pretvaranja boja slike u nijanse sive te binarizacije slike. U ovom potpoglavlju predstavljeni su rezultati primjene algoritama predstavljenih u poglavlju 2.1.1 i 2.1.2 dok je opis implementacije rješenja dan u poglavlju 6.



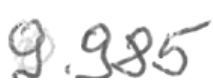



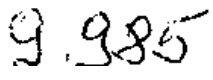


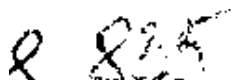
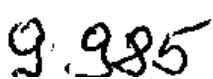
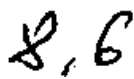

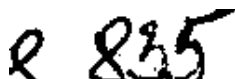
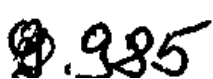


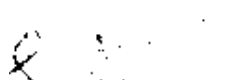
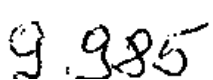


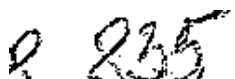

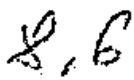





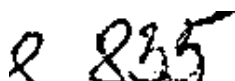
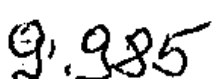
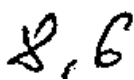

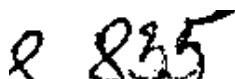

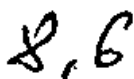












Tablica 1 prikazuje primjenu algoritama pretvaranja boja slike u nijanse sive prilikom korištenja plave, crvene i crne kemijske te grafitne olovke za pisanje broja „7.771“.

Tablica 1. Rezultati primjene algoritama pretvaranja slike u nijanse sive

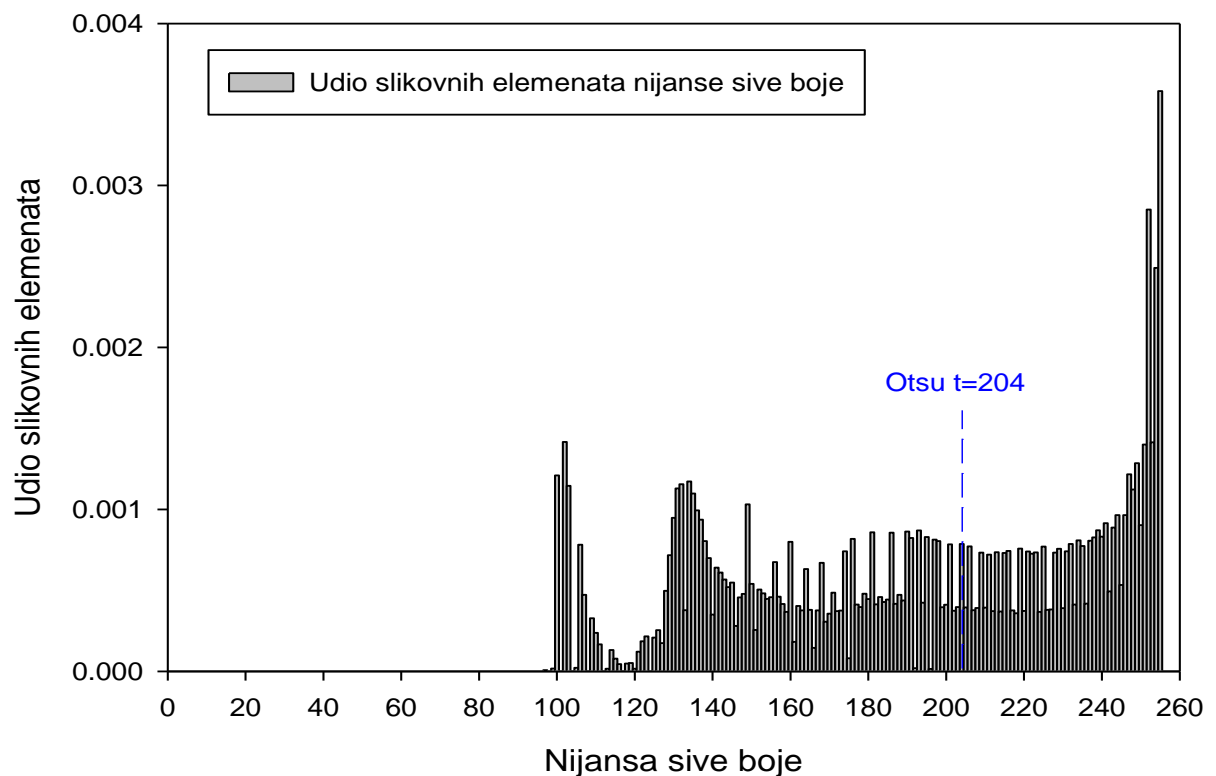
	plava kemijska	crvena kemijska	crna kemijska	grafitna olovka
original				
prosjeck inteziteta				
kolorimetrijska metoda				
jednokanalna metoda (crvena)				
jednokanalna metoda (zelena)				
jednokanalna metoda (plava)				
maksimum inteziteta				
minimum inteziteta				
uklanjanje zasićenosti				

Tablica 2 prikazuje primjenu algoritama binarizacije na skenirane obrasce. Binarizacija je provedena na čitavom obrascu, a unutar tablice su prikazani odabrani dijelovi obrasca kao što su markeri te nekoliko brojeva. Broj „8.835“ je odabran jer je svjetliji od ostatka obrasca, broj „9.985“ je odabran jer sadrži prebrisanu prvu znamenku 9, a broj „8.6“ je broj koji nije imao dodatnih posebnosti kao i većina mogućih uzoraka.

Tablica 2. Primjeri rezultata primjene algoritama binarizacije

original				
fiksni prag (200)				
fiksni prag (230)				
fiksni prag (250)				
Otsu metoda				
Niblack ( $k = -0.2, r = 1$ )				
Niblack ( $k = -0.2, r = 3$ )				
Niblack ( $k = -0.2, r = 6$ )				
Niblack ( $k = -0.2, r = 10$ )				
Sauvola ( $k = 0.1, r = 3$ )				
Sauvola ( $k = 0.05, r = 3$ )				
Sauvola ( $k = 0.1, r = 10$ )				

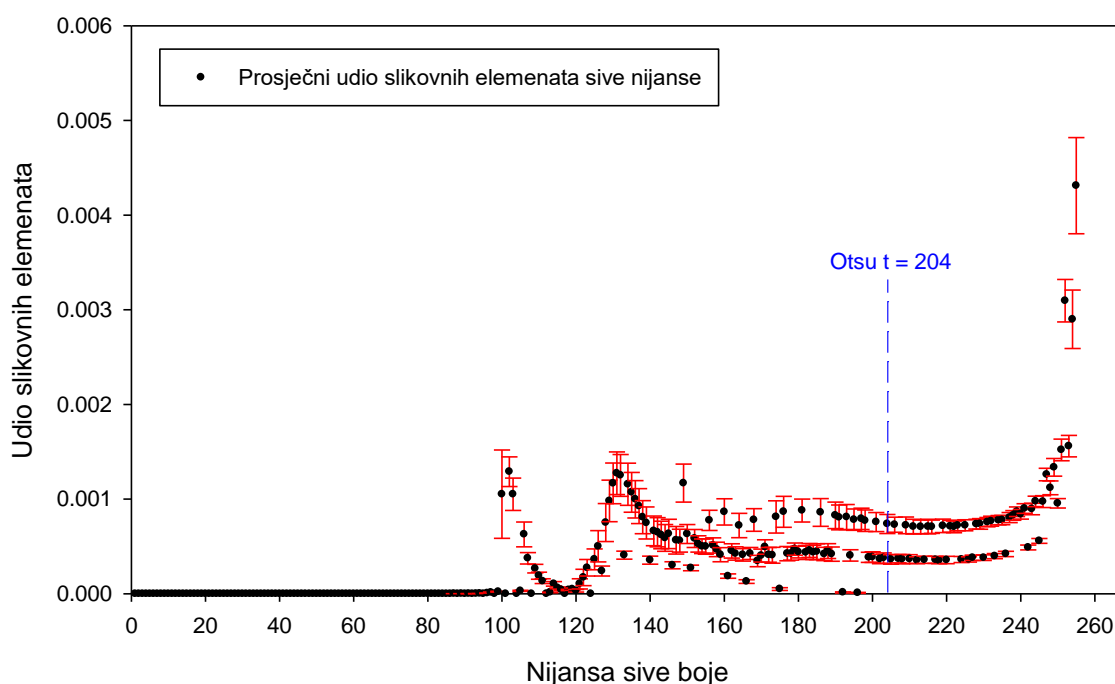
Kako bi prikazali problem s kojim se suočavaju algoritmi binarizacije na grafu 6 dan je histogram nijansi sive boje dobiven na temelju obrasca korištenog za generiranje rezultata binarizacije u tablici 2, a na grafu 7 dan je prosječni histogram obrazaca skeniranih u okviru ovog završnog rada. Vrijednost 255 (bijela boja) je izuzeta iz grafova jer zauzima 90.3% slike u prosjeku, odnosno 90.6% na trenutno razmatranom obrascu. Kako je metoda Otsu, metoda globalne binarizacije na grafovima je dodana dodatna oznaka za prag koji je metoda Otsu postavljala.



Graf 6. Udio slikovnih elemenata prema nijansi sive boje promatranog obrasca

U nastavku ovog rada odabrani su algoritam minimuma intenziteta za algoritam pretvaranja boja u nijanse sive te algoritam fiksnog praga uz prag jednak 250 za binarizaciju obrazaca. Algoritam fiksnog praga uz prag jednak 250 je odabran zbog veće brzine izračuna u odnosu na ostale metode koje su postizale sličnu kvalitetu binarizacije slike.





Graf 7. Prosječni udio slikovnih elemenata prema nijansi sive boje

### 3.3. Određivanje orijentacije slike i izdvajanje polja s bodovima studenata

Analizom skeniranih obrazaca izmjerena je prosječna zarotiranost od  $-0.28^\circ$  u odnosu na predložak, a maksimalna zarotiranost je iznosila  $-0.42^\circ$ . Rotaciju smo obavljali koristeći teoretsku podlogu predstavljenu u poglavlju 2.1.3, a u nastavku je dan primjer postupka transformacije za skenirani obrazac.

Slika 15. predstavlja početno stanje prilikom prilagođavanja orijentacije obrasca pri čemu je na lijevoj strani slike dan predložak obrasca, a na desnoj strani slike se nalazi skenirani obrazac. Položaji oznaka na predlošku obrasca iznose  $O(114, 132)$ ,  $X(1491, 132)$ ,  $Y(114, 2189)$ , a položaji oznaka na skeniranom obrascu iznose  $O_1(117, 119)$ ,  $X_1(1499, 127)$ ,  $Y_1(107, 2169)$ .

0		8.9		24549	
1		9.0		23061	
2		13.14		6059	
3		70.97		2058	
4		0.501		3.03	
5		1.767		8.07	
6		2.818		1.08	
7		3.484		9.09	
8		4.955		0.24	
9		5.152		6.33	
12		6.99		2.0	
34		7.771		4.38	
56		8.835		5.96	
78		9.985		7.90	
90		10.005		8.6	
31		16.219		5.43	
42		446.17		10.22	
53		89.626		3.45	
64		28.65		6.8	
75		73869		12.3	
0.1		727.802		40.5	
1.2		4041		9.88	
2.3		4374		7.65	
3.4		7922		14.73	
4.5		33291		92.60	
5.6		8257		0.636	
6.7		6687		11.96	
7.8		93947		51.88	

0	0	8.9	8.9	24549	24549
1	1	9.0	9.0	23061	23061
2	2	13.14	13.14	6059	6059
3	3	70.97	70.97	2058	2058
4	4	0.501	0.501	3.03	3.03
5	5	1.767	1.767	8.07	8.07
6	6	2.818	2.818	1.08	1.08
7	7	3.484	3.484	9.09	9.09
8	8	4.955	4.955	0.24	0.24
9	9	5.152	5.152	6.33	6.33
12	12	6.99	6.99	2.0	2.0
34	34	7.771	7.771	4.38	4.38
56	56	8.835	8.835	5.96	5.96
78	78	9.985	9.985	7.90	7.90
90	90	10.005	10.005	8.6	8.6
31	31	16.219	16.219	5.43	5.43
42	42	446.17	446.17	10.22	10.22
53	53	89.626	89.626	3.45	3.45
64	64	28.65	28.65	6.8	6.8
75	75	73869	73869	12.3	12.3
0.1	0.1	727.802	727.802	40.5	40.5
1.2	1.2	4041	4041	9.88	9.88
2.3	2.3	4374	4374	7.65	7.65
3.4	3.4	7922	7922	14.73	14.73
4.5	4.5	33291	33291	92.60	92.60
5.6	5.6	8257	8257	0.636	0.636
6.7	6.7	6687	6687	11.96	11.96
7.8	7.8	93947	93947	51.88	51.88

Slika 15. Usporedba uokvirenog predloška obrasca i uokvirenog skeniranog obrasca

U okviru ovog primjera razmatrat ćemo položaj prve ćelije za unos bodova. Pozicije prve ćelije za unos bodova definirat ćemo točkom gornjeg lijevog ruba ćelije te točkom donjeg desnog ruba ćelije. Pozicija ćelije na predlošku obrasca definirana je točkama  $P_1(343, 246)$ ,  $P_2(617, 309)$ , a na skeniranom obrascu točkama  $K_1(347, 236)$ ,  $K_2(621, 301)$ .

Cilj određivanja orijentacije je poznajući pozicije oznaka na predlošku i skeniranom obrascu te pozicije ćelija na predlošku odrediti poziciju ćelija na skeniranom obrascu  $K_1, K_2$ . Prvi korak određivanja pozicija  $K_1, K_2$  je odrediti transformacije  $T_1, T_2, T_3, T_4$  definirane u 1.3, a izračun za predstavljeni primjer dan je u nastavku.

- Transformacija translacije  $T_1$ .

$$T_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x_{o1} & -y_{o1} & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -117 & -119 & 1 \end{bmatrix}$$

- Transformacija rotacije  $T_2$ .

$$\vec{i} = \frac{\overrightarrow{OX}}{|\overrightarrow{OX}|} = \vec{u}, \vec{j} = \frac{\overrightarrow{OY}}{|\overrightarrow{OY}|} = \vec{v}$$

$$\vec{i}_1 = \frac{\overrightarrow{OX_1}}{|\overrightarrow{OX_1}|} = 0.9999\vec{u} + 5 \cdot 10^{-3}\vec{v}, \quad \vec{j}_1 = \frac{\overrightarrow{OY_1}}{|\overrightarrow{OY_1}|} = -4.8780\vec{u} + 0.99999\vec{v}$$

$$\cos(\alpha) = \vec{l}_1 \cdot \vec{l} = 0.999983$$

$$|\sin(\alpha)| = |\vec{l}_1 \times \vec{l}| = 5.7886 \cdot 10^{-3}$$

$$g(y_X = 2189, y_{X1} = 2169) = -1$$

$$T_2 = \begin{bmatrix} \cos(\alpha) & \sin(\alpha g(y_X, y_{X1})) & 0 \\ -\sin(\alpha g(y_X, y_{X1})) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$T_2 = \begin{bmatrix} 0.999983 & -5.7886 \cdot 10^{-3} & 0 \\ 5.7886 \cdot 10^{-3} & 0.999983 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Transformacija skaliranja  $T_3$ .

$$s_x = \frac{|OX|}{|OX_1|} = 0.99637, \quad s_y = \frac{|OY|}{|OY_1|} = 1.00340$$

$$T_3 = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.99637 & 0 & 0 \\ 0 & 1.00340 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Transformacija translacije  $T_4$ .

$$T_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x_o & y_o & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 114 & 132 & 1 \end{bmatrix}$$

Konačna matrica transformacije dana je sljedećim izrazom

$$T = T_1 T_2 T_3 T_4 = \begin{bmatrix} 0.99635 & -5.8083 \cdot 10^{-3} & 0 \\ 5.7676 \cdot 10^{-3} & 1.0033 & 0 \\ -3.25965 & 13.277 & 1 \end{bmatrix}$$

Točke  $P_1$  i  $P_2$  transformiramo u homogene koordinate te one zatim iznose  $P_{1h}(343, 246, 1), P_{2h}(617, 309, 1)$ .

Idući korak je odrediti transformirane  $P_{1h}'$  i  $P_{2h}'$ . Izračun transformiranih koordinata dan je u nastavku.

$$P_{1h}' = P_{1h} T^{-1} = [346.174 \quad 233.942 \quad 1]$$

$$P_{2h}' = P_{2h} T^{-1} = [620.805 \quad 298.32 \quad 1]$$

Točke  $P_{1h}', P_{2h}'$  zatim transformiramo iz homogenog prostora u radni prostor te one iznose  $P_1'(346.174, 233.942), P_2'(620.805, 298.32)$ . U konačnici možemo usporediti izračunate zaokružene vrijednosti pozicija ćelije na skeniranom obrascu  $P_1'(346, 233), P_2'(621, 298)$  i izmjerenih vrijednosti  $K_1(347, 236), K_2(621, 301)$ . Vrijednosti koordinata točaka  $P_1, P_2$  su

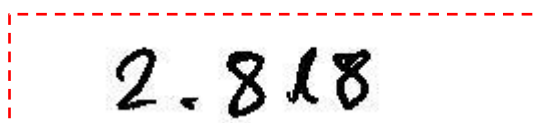
zaokružene jer sliku možemo indeksirati s cjelobrojnomo vrijednošću. Odstupanja pojedinih koordinata dana su u nastavku.

$P_1' - K_1'$	$ \Delta x_1  = 1$	$ \Delta y_1  = 3$
$P_2' - K_2'$	$ \Delta x_2  = 0$	$ \Delta y_2  = 3$

Uzroci greške su odstupanja prilikom određivanja centara oznaka, odstupanja prilikom određivanja rubnih točaka ćelije te činjenica da je slika diskretizirana.

Nakon što smo odredili koordinati sustav obrasca i prilagodili njegovu orijentaciju osnovnom obrascu možemo izdvojiti ćelije tablice koje sadrže bodove studenata. Korisnik je prilikom definiranja obrasca definirao ćelije obrasca pomoću informacija o gornjem lijevom kutu ćelije te visine i širine ćelije. Način unosa informacija o ćelijama u implementiranom sustavu opisan je u poglavlju 6.

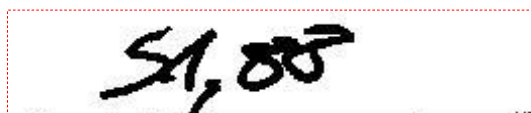
Na slici 16 dan je primjer izdvojene ćelije s bodovima studenata koja je uokvirena isprekidanom crvenom linijom kako bi se sačuvao prikaz praznog područja regije.



Slika 16. Primjer izrezane regije

Na krajnje desnim i donjim područjima tablice je moglo doći do pogreške prilikom rezanja zbog navedenih uzroka greške prilikom određivanja pozicije ćelije.

Primjer greške prilikom izrezivanja regije dan je na slici 17. Grešku smo smanjivali tako što smo odrezali nekoliko slikovnih elemenata sa svake strane ćelije.



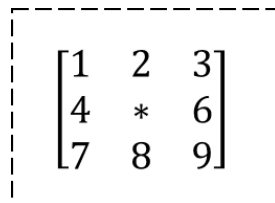
Slika 17. Primjer greške prilikom izrezivanja regije s bodovima studenata

## 4. Segmentacija znamenki

U okviru ovog poglavlja dan je algoritam označavanja povezanih komponenti korišten u segmentaciji znamenki broja te dobiveni rezultati i problemi s kojima se algoritam susreće.

### 4.1. Algoritam označavanja povezanih komponenti

Označavanju povezanih komponenti možemo pristupiti na više načina. Algoritam označavanja povezanih komponenti korišten u okviru ovog rada je preuzet iz [10] i koristi susjedstvo od osam slikovnih elemenata. Susjedstvo od osam slikovnih elemenata prikazano je na slici 18 pri čemu (\*) označava trenutni slikovni element, a brojevi od 1 do 8 predstavljaju susjedne slikovne elemente.



Slika 18. Prikaz susjedstva od 8 slikovnih elemenata

Ulaz algoritma je binarizirana slika koja sadrži znamenke broja, a izlaz su označene komponente koje će biti kandidati za znamenke. Korišteni pseudokod algoritma i pomoćnih metoda dan je u nastavku.

Inicijalizacija globalnih podatkovnih struktura

```
procedura initialize():  
    label = 0  
    za i= 1 do MaxLab ponavljaj:  
        parent[i] = 0
```

Algoritam 6. Označavanje povezanih komponentata (prilagođen iz [10]) – inicijalizacija

Varijabla `label` predstavlja trenutno dodijeljenu oznaku, a niz `parent` predstavlja niz čiji element `parent[i]` predstavlja roditeljsku oznaku oznake `i`. Varijabla `MaxLab` je jednaka umnošku visine i širine slike te predstavlja maksimalnu moguću oznaku.

### Izračun povezanih komponenti

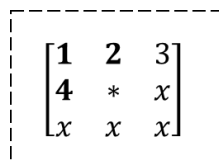
B – originalna slika, LB – označene povezane komponente

```
procedura find_components(B, LB)
    Initialize()
    //prvi prolaz - dodjeljivanje inicijalnih oznaka
    za L=0 do MaxRow ponavljaaj:
        //inicijalizacija oznaka trenutnog reda na 0
        za P=0 do MaxCol ponavljaaj:
            LB[L,P]=0
        //dodjela oznaka
        za P=0 do MaxCol ponavljaaj:
            ako B[L,P] == 1:
                A = prior_neighbors(L,P)
                ako prazan(A):
                    {M:=label; label:=label+1;}
                inače M:=min(labels(A));
                LB[L,P]:=M;
                za svaki X iz labels(A) i X!=M ponavljaaj:
                    union(M, X, PARENT);

    //drugi prolaz - zamjena ekvivalentnih oznaka
    za L=0 do MaxRow ponavljaaj:
        za P=0 do MaxCol ponavljaaj:
            if B[L,P] == 1:
                LB[L,P] = find(LB[L,P], parent)
```

Algoritam 7. Označavanje povezanih komponenata (prilagođen iz [10])

Procedura `find_components` je glavna procedura algoritma koja se sastoji od dva dijela. Prvi dio je prolaz kroz sliku i dodjela inicijalnih oznaka komponentama. Slikovnom elementu dodjeljujemo oznaku susjedne ćelije kojoj smo dali najmanju oznaku. Susjedne ćelije dohvaćamo metodom `prior_neighbors` koja vraća susjedne ćelije koje smo do sada obradili. Slika 19 prikazuje susjedstvo od 8 slikovnih elemenata s istaknutim susjedima kojima smo dodijelili oznake te je takav poredak dodjele oznaka rezultat redoslijeda obilaska slike.



Slika 19. Susjedstvo od 8 slikovnih elemenata s označenim susjedima

Drugi dio procedure zamjenjuje ekvivalentne oznake najmanjim oznakama. Zamjena ekvivalentnih oznaka je potrebna zbog načina obilaska slike. Ostale pomoćne procedure korištene u algoritmu su prikazane u algoritmu 8.

```
//traži izvorišnu oznaku oznake X pretragom roditelja
procedura find (X, parent):
    j=X
    dok parent[j] !=0 ponavljaaj
        j = parent[j]
    vrati j

//procedura radi uniju dviju oznaka X i Y te ažurira polje parent
procedura union (X, Y, parent):
    j=X
    k=Y
    dok parent[j] != 0 ponavljaaj:
        j = parent[j]
    dok parent[k] != 0 ponavljaaj:
        k = parent[k]
    ako j!=k:
        parent[k]=j
```

Algoritam 8. Označavanje povezanih komponenata (prilagođen iz [10]) – pomoćne metode

## 4.2. Filtriranje segmenata

Filtriranje segmenata je postupak odbacivanja suvišnih segmenata. Postupak uvodimo zbog problema prezentiranog u poglavlju 2.2 gdje je na slici 8 skup slikovnih elemenata koji ima crvenu boju suvišan jer je nastao kao rezultat šuma. Takvi slikovni elementi mogu nastati prilikom skeniranja zbog diskretizacije, mogućom prašinom na skeneru te kao posljedica postupka binarizacije. Algoritam Niblack je jedan primjer algoritma binarizacije koji može

prouzročiti neželjene slikovne elemente kao što je prikazano u tablici 2. Drugi problem koji želimo umanjiti je problem prezentiran u poglavlju 3.3 na slici 17 gdje želimo odbaciti segmente koji pripadaju granici ćelije ako ih možemo odvojiti od znamenaka.

U okviru ovog rada predstavljena su dva postupka filtriranja korištena prilikom analize.

- Prvi postupak filtriranja segmenata.

Postupak se temelji na odbacivanju empirijski određenih premalih slikovnih elemenata. Za slike u rezoluciji od 200 DPI vrijedile su sljedeće vrijednosti. Prepoznata komponenta koja je imala manje od 3 slikovna elementa širine ili visine ili je imala ukupan broj točaka manji od 15 se odbacivala. Vrijednosti su se skalirale prema veličini, odnosno iznosu rezolucije obrasca. Ova metoda je postizala rezultat od 78.95 % uspješne segmentacije, 2 % prekomjerne segmentacije te 19.05% nedovoljne segmentacije.

Problem kod ovog postupka je bio što je osim malih segmenata uglavnom odbacivao i decimalne točke i decimalne zareze.

- Drugi postupak filtriranja segmenata.

Postupak segmentiranja je proširen podjelom broja na regije. Kako bi reprezentirali područje u kojoj je većina slikovnih elemenata broj ćemo ograničiti s dvije linije kao što je prikazano na slici 20.



Slika 20. Prikaz horizontalne projekcije te gornje i donje osnovne linije (preuzeto iz [21])

Prva linija određuje gornji rub većine slikovnih elemenata broja (engl. *upper baseline*), a druga linija određuje donji rub većine slikovnih elemenata broja (engl. *lower baseline*). Određivanje tih dviju linija je preuzeto iz [21] te je dano u nastavku. Prvi korak je određivanje horizontalne projekcije  $P_h$  binarizirane slike te određivanje  $y$  koordinate na slici s najvećim intenzitetom  $P_{max}$ .

$$P_{max} = \max_y P_h(y)$$

$$y_{P_{max}} = \max_y y \in \{y | P_h(y) = P_{max}\}$$

Zatim gornju  $y_{up}$  i donju  $y_{low}$  liniju određujemo sljedećim izrazima.



$$y_{up} = \min_y y \in \{y | P_h(y) \leq \frac{1}{4} P_{max}, y_{P_{max}} < y \leq y_{max}\}$$

$$y_{low} = \max_y y \in \{y | P_h(y) \leq \frac{1}{3} P_{max}, 0 < y \leq y_{P_{max}}\}$$

Odbacivanje segmenata zatim obavljammo odbacivanjem sljedećih regija.

1. Regije koje su između  $y_{up}$  i vrha slike, te regije koje su između  $y_{low}$  i dna slike, a po širini se prostiru na prostor veći od trećine širine slike.
2. Odbacujemo i regije koje su u od lijevog kraja slike udaljeni 5 slikovnih elemenata, odnosno od desnog kraja slike udaljeni 5 slikovnih elemenata, a po visini veći od trećine visine slike.
3. Regije koje imaju manje od 4 slikovnih elemenata.
4. Regije između linija  $y_{low}$  i  $y_{up}$  koje imaju širinu ili visinu manju od 2 slikovna elementa.

Takav postupak postiže stopu od 78.51 % uspješne segmentacije, 7.47% prekomjerne segmentacije te 14.02% nedovoljne segmentacije. Ovaj postupak je bolje segmentirao decimalne točke i zareze, no kako je imao manju granicu na ukupan broj točaka regije imao je veću stopu prekomjerne segmentacije.

Opisani postupci filtriranja segmenata su ovisni o tome da nam je poznata informacija o odnosu veličine predloška obrasca i veličine skeniranog obrasca.

### 4.3. Rezultati segmentacije

Ulazni skup za segmentaciju su bila polja 22 obrasca, od kojih je svaki obrazac sadržavao 84 polja koja su sadržavala međusobno različite brojeve. Ukupan broj polja je iznosio 1848 polja. U tablici 3 su prikazani rezultati segmentacije. Zasebno su prikazani stopa uspješne segmentacije, stopa prekomjerne segmentacije i stopa nedovoljne segmentacije. Stopa uspješne segmentacije je prikazana po istovjetnosti broja očekivanih znakova zbog jednostavnosti tehnike, a stvarna vrijednost je manja ili jednaka ovisno o tome da li su segmenti zadovoljavajući.

Tablica 3. Rezultati segmentacije po brojevima

Originalni broj	Stopa uspješne segmentacije	Stopa prekomjerne segmentacije	Stopa nedovoljne segmentacije
0	100%	0%	0%
1	95.45%	4.55 %	0%
2	90.91 %	9.09%	0%
3	100%	0%	0%
4	90.91 %	9.09%	0%
5	81.82%	18.18%	0%
6	90.91 %	9.09%	0%
7	95.45%	4.55 %	0%
8	100%	0%	0%
9	95.45%	4.55 %	0%
12	81.82%	18.18%	0%
34	100%	0%	0%
56	95.45%	4.55 %	0%
78	68.18%	9.09%	22.73%
90	77.27%	4.55%	18.18%
31	95.45%	4.55 %	0%
42	95.45%	4.55 %	0%
53	86.36%	4.55%	9.09%
64	100%	0%	0%
75	86.36%	4.55%	9.09%

Originalni broj	Stopa uspješne segmentacije	Stopa prekomjerne segmentacije	Stopa nedovoljne segmentacije
0.1	100%	0%	0%
1.2	95.45%	4.55 %	0%
2.3	90.90 %	4.55%	4.55%
3.4	100%	0%	0%
4.5	81.82%	9.09%	9.09%
5.6	90.91 %	9.09%	0%
6.7	90.91 %	9.09%	0%
7.8	100%	0%	0%
8.9	81.82%	0%	18.18%
9.0	81.82%	13.63%	4.55%
13.14	81.82%	0%	18.18%
70.97	31.81%	13.63%	54.56%
0.501	77.27%	0%	22.73%
1.767	54.55%	0%	45.45%
2.818	77.27%	4.55%	18.18%
3.484	81.82%	0%	18.18%
4.955	72.27%	9.09%	18.64%
5.152	54.55%	13.63%	31.82%
6.99	90.91 %	9.09%	0%
7.771	72.27%	4.55%	23.18%
8.835	77.27%	4.55%	18.18%
9.985	68.18%	18.19%	13.63%
10.005	54.55%	4.55%	40.9%
16.219	72.27%	4.55%	23.18%
446.17	72.27%	0%	27.73%
89.626	68.18%	0%	31.82%
28.65	68.18%	13.63%	18.19%
73869	63.63%	13.63%	22.74%
727.802	50%	4.55%	45.45%
4041	86.36%	0%	13.64%

Originalni broj	Stopa uspješne segmentacije	Stopa prekomjerne segmentacije	Stopa nedovoljne segmentacije
4374	81.82%	0%	18.18%
7922	63.64%	9.09%	27.27%
33291	81.82%	9.09%	9.09%
8257	54.55%	9.09%	36.36%
6687	77.27%	4.55%	18.18%
93947	90.91 %	9.09%	0%
24549	72.72%	13.64%	13.64%
23061	77.27%	0%	22.73%
6059	63.63%	4.55%	31.82%
2058	36.37%	13.64%	50.01%
3,03	95.45%	0%	4.55%
8,07	77.27%	9.09%	13.64%
1,08	77.27%	13.64%	9.09%
9,09	81.82%	9.09%	9.09%
0,24	90.91 %	0%	9.09%
6,33	86.36%	4.55%	9.09%
2,0	68.18%	4.55%	27.27%
4,38	81.81%	4.55%	13.64%
5,96	72.27%	9.09%	18.64%
7,90	77.27%	13.64%	9.09%
8,6	86.36%	9.09%	4.55%
5,43	86.36%	13.64%	0%
10,22	63.63%	13.63%	22.74%
3,45	72.27%	9.09%	18.64%
6,8	90.90 %	4.55%	4.55%
12,3	77.27%	18.18%	4.55%
40,5	72.72%	13.64%	13.64%
9,88	54.55%	13.63%	31.82%
7,65	63.64%	27.27%	9.09%
14,73	59.09%	27.27%	13.64%

Originalni broj	Stopa uspješne segmentacije	Stopa prekomjerne segmentacije	Stopa nedovoljne segmentacije
92,60	63.64%	18.18%	18.18%
0,636	72.72%	13.64%	13.64%
11,96	59.09%	22.73%	18.18%
51,88	45.46%	18.18%	36.36%

Na tablici 3 zelenom bojom su označeni brojevi koji su u potpunosti točno segmentirani, a crvenom bojom su označeni brojevi za koje algoritam ima stopu uspješnosti manju od 50%.

Sažeti prikaz dan je na tablici 4 koji grupira brojeve prema broju znakova.

Tablica 4. Rezultati segmentacije grupirani po broju znakova broja

Broj znakova	Ukupan udio primjera	Stopa uspješne segmentacije	Stopa prekomjerne segmentacije	Stopa nedovoljne segmentacije
1	12%	94.09%	5.91%	0%
2	12%	88.63%	5.0%	6.36%
3	15%	89.16%	5.25%	5.60%
4	27%	74.90%	8.5%	16.50%
5	27%	68.18%	10.48%	21.34%
6	5%	67.05%	2.28%	30.68%
7	1%	50%	4.55%	45.45%

Ukupna stopa uspješne segmentacije iznosi 78.51%, ukupna stopa prekomjerne segmentacije iznosi 6.27% te ukupna stopa nedovoljne segmentacije iznosi 15.22%.

Uspješnost je mjerena prema broju izdvojenih segmenata.

## 5. Klasifikacija znamenki

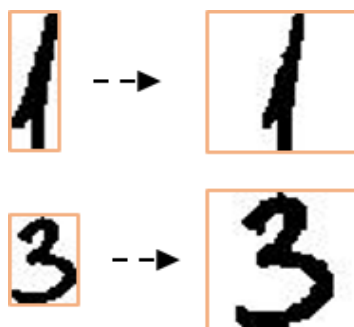
U okviru ovog poglavlja opisan je rad s unaprijednom slojevitom neuronskom mrežom za klasifikaciju znamenaka te rezultati koje je postizala na ulaznom skupu primjera.

Ulazni skup primjera se sastojao od 12 znakova (10 dekadskih znamenki, decimalni zarez i decimalna točka) podijeljenih u 11 klasa. Decimalni zarez i točku smo reprezentirali jednom klasom zbog sličnosti prilikom normalizacije veličine slike, a i zato što nam ih nije potrebno razlikovati. Ukupno je korišteno 6092 primjera, a ukupan broj primjera po svakom znaku je bio u rasponu od 471 do 534 primjera, od toga je 60% skup za učenje, 20% skup za provjeru te 20% skup za testiranje. Ako skup primjera za neki znak nije mogao biti podijeljen na navedene omjere bez ostatka, ostatak se prebacivao u skup za testiranje.

### 5.1. Prilagodba ulaznih podataka

Znamenke dobivene segmentacijom brojeva su različitih dimenzija, a kako neuronska mreža ima konstantan broj ulaznih neurona znamenke ćemo morati prilagoditi. Postupak koji ćemo koristiti zove se centriranje i normalizacija veličine slike (engl. *position and size normalization, bounding box normalization and centering*) koja je korištena i u najpoznatijoj bazi primjera znamenki MNIST [18], a definicija postupaka je preuzeta iz [19] i [20]. Osnovne dimenzije slike u koju ćemo stavljati znamenke će biti  $28 \times 28$  kao što je korišteno i u MNIST bazi.

Cilj normalizacije veličine slike bit će skalirati veću stranicu slike znaka na željenu veličinu, a drugu stranicu skalirati proporcionalno tako da se zadrži odnos stranica početne slike znaka. Skaliranu sliku ćemo zatim postaviti u centar. Algoritam primjene opisanog postupka dan je u algoritmu 9. Primjer normalizacije slike dan je na slici 21.



Slika 21. Primjer normalizacije slika segmenata

```

funkcija prilagodiUlaz (slika s, int dim):
    maxDim  $\leftarrow$  max(s.sirina, s.visina)
    faktor  $\leftarrow \frac{dim}{maxDim}$  //faktor skaliranja
    novaVisina  $\leftarrow$  s.visina*faktor
    novaSirina  $\leftarrow$  s.sirina*faktor
    pomakX  $\leftarrow \frac{dim-novaSirina}{2}$ 
    pomakY  $\leftarrow \frac{dim-novaVisina}{2}$ 

    novaSlika  $\leftarrow$  Slika(novaSirina, novaVisina)
    novaSlika.nacrtaj(s, pomakX, pomakY, 0, 0, novaSirina + pomakX,
novaVisina + pomakY)
    //pretpostavka crtanje skalira sliku na dane dimenzije

    vрати novaSlika

```

#### Algoritam 9. Normalizacija veličine slike znaka

Nakon što smo proveli normalizaciju veličine slike potrebno je sliku pretvoriti u jednodimenzionalno polje kakvo prima neuronska mreža. Svaka pozicija u polju predstavlja ulaz jednog neurona. Pretvorba slike u polje dana je u algoritmu 10.

```

funkcija pretvoriUPolje (Slika s):
    ulaznoPolje = polje[s.sirina*s.visina]

    za svaki slikovni element(x,y) slike s ponavljaj:
        ulaznoPolje[y*s.visina+x] = s[Tocka(x,y)]

    vрати ulaznoPolje

```

#### Algoritam 10. Pretvorba slike u jednodimenzionalno polje

## 5.2. Arhitektura neuronske mreže

Korištena neuronska mreža se sastojala od tri sloja. Prvi, ulazni, sloj se sastojao od 784 neurona što odgovara slikama veličine  $28 \times 28$  slikovnih elemenata. Skrivenom sloju smo podešavali broj neurona. Treći, izlazni, sloj se sastojao od 11 neurona što odgovara broju klasa u koje želimo klasificirati ulazne primjere.

Skrivenom sloju smo inicijalno postavili broj neurona na 15, kao što je predloženo u [22] gdje su radili klasifikaciju znamenki iz MNIST baze rukom pisanih znakova. Rezultat na skupu za testiranje je iznosio 80.28% uspješno klasificiranih primjera na skupu za testiranje.

Kada nije bilo skrivenog sloja, neuronska mreža je postizala rezultat od 82.41% uspješno klasificiranih primjera skupa za testiranje koji se sastojao od 1217 znakova.

Za podešavanje broja neurona skrivenog sloja fiksirali smo stopu učenja na 0.05, te smo zatim tražili najmanji broj neurona za koje je neuronska mreža bolje klasificirala. Stopa učenja je određena tako što se pokazalo da pri prve dvije testirane konfiguracije neuronske mreže neuronska mreža uspijeva učiti klasificirati ulazne primjere. Broj neurona smo htjeli smanjiti kako bi povećali mogućnost generalizacije neuronske mreže, dok smo broj neurona povećavali kako bi povećali kapacitet neuronske mreže da klasificira ulazne primjere.

Očekivani izlaz iz izlaznog sloja smo predstavili kao niz s 11 znakova čiji su elementi svi osim jednog jednaki nuli, a element koji predstavlja broj klase jednak jedan. Klasa ulaznog primjera je određivana tako što je izračunato koji neuron daje maksimalni izlaz. Neuron koji je davao maksimalni izlaz predstavlja klasu za čiji ulazni primjer ima najveću pripadnost.

Konačno je korištena neuronska mreža s dva sloja. Prvi sloj je bio ulazni sloj sa 784 ( $28 \times 28$ ) neurona, a drugi sloj je bio izlazni sloj s 11 neurona.

Poznavajući arhitekturu neuronske mreže može se odrediti iznos maksimalne vrijednosti kriterijske funkcije  $E$ . Maksimum kriterijske funkcije se postiže kada svi izlazni neuroni za koje je očekivani izlaz 0 daju na izlazu 1 i kada izlazni neuron za koji očekujemo izlaz 1 daje 0. Tada će maksimalni iznos kriterijske funkcije iznositi

$$E = \frac{1}{2N} \sum_{s=1}^N \sum_{i=1}^m (t_{s,i} - o_{s,i})^2 = \frac{1}{2N} Nm(1 - 0)^2 = \frac{1}{2}m = 5.5$$



### 5.3. Rezultati klasifikacije

Koristeći arhitekturu mreže opisanu u poglavlju 5.2 postizali smo uspješnost klasifikacije od 82.41% uspješno klasificiranih primjera na skupu za testiranje koji se sastojao od 1217 znakova. Uspješnost na cjelokupnom skupu primjera znakova je iznosio 87% od 6092 znaka.

Skup korištenih znakova predstavlja skup znakova koji su bili uspješno segmentirani.

Kako bi bolje opisali rezultat ispravno i neispravno klasificiranih ulaznih primjera rezultati klasifikacije su prikazani u tablici 5. koja prikazuje matricu zabune klasifikacije određene znamenke. Redci tablice označavaju očekivanu klasifikaciju, a stupci tablice prikazuju rezultat klasifikacije neuronske mreže. Pojedina ćelija prikazuje udio klasificiranih primjera znaka označenog u retku u klasu označenu u stupcu.

Tablica 5. Rezultat klasifikacije prikazan matricom zabune

	0	1	2	3	4	5	6	7	8	9	. ili ,
0	0.95	0.0	0.0	0.0	0.01	0.0	0.0	0.0	0.0	0.01	0.03
1	0.02	0.80	0.0	0.0	0.02	0.0	0.0	0.0	0.01	0.05	0.1
2	0.01	0.0	0.81	0.01	0.03	0.02	0.0	0.01	0.03	0.04	0.04
3	0.02	0.0	0.0	0.81	0.0	0.02	0.0	0.05	0.01	0.07	0.03
4	0.0	0.06	0.0	0.0	0.80	0.02	0.05	0.04	0.0	0.0	0.03
5	0.01	0.01	0.0	0.02	0.03	0.83	0.06	0.0	0.01	0.01	0.01
6	0.01	0.01	0.01	0.0	0.02	0.04	0.99	0.0	0.0	0.0	0.01
7	0.02	0.01	0.01	0.0	0.02	0.01	0.01	0.86	0.01	0.0	0.04
8	0.02	0.05	0.06	0.04	0.01	0.04	0.04	0.01	0.58	0.06	0.07
9	0.03	0.01	0.0	0.04	0.04	0.02	0.0	0.0	0.04	0.80	0.03
. ili ,	0.01	0.09	0.0	0.003	0.0	0.02	0.0	0.003	0.004	0.0	0.87

Na slici 22 dan je primjer nekoliko krivo klasificiranih uzoraka. Točne klasifikacije za dane uzorke slijedno idu: „8“, „9“, „1“.



Slika 22. Primjer krivo klasificiranih znamenki. 8 je klasificirano kao 6, 9 je klasificirano kao 3 i 1 je krivo klasificirano kao decimalni separator

Temeljem slike 22 možemo zaključiti kako bi za unaprjeđenje postupka klasifikacije trebalo unaprijediti postupak normalizacije slika znakova koje sadrže decimalne separatore. Decimalne separatori su specifični po tome što im je veličina manja od prosječne veličine znakova unutar slike koja sadrži broj bodova studenta.

Konačni rezultat ispravno prepoznatih višeznamenkastih brojeva u odnosu na čitav ulazni skup podataka dan je u poglavlju 7.

## 6. Implementacija sustava

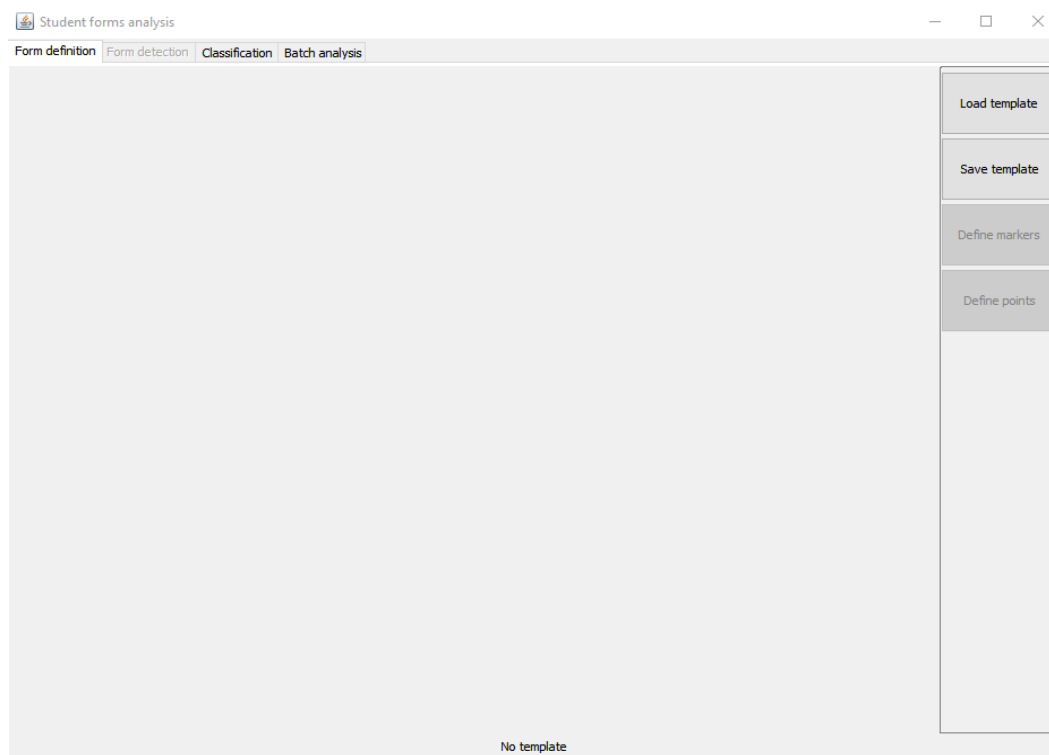
U okviru ovog završnog rada implementiran je programski sustav za lokalizaciju i očitavanja rukom pisanih bodova studenta koristeći programski jezik Java verzije 1.8.

U idućim potpoglavljima opisan je izgrađeni programski sustav.

### 6.1.1. Grafičko korisničko sučelje

Grafičko korisničko sučelje se sastoji od četiri prikaza unutar jednog prozora. Prikazi su dani u nastavku:

- Prikaz za definiciju predloška formulara. Prikaz omogućava korisniku da definira novi ili učitava postojeći predložak obrasca. Slika 23 prikazuje stanje prikaza prije nego što korisnik učitava predložak. Prikaz se sastoji od trake za odabir prikaza, alatne trake za odabir akcije nad učitanim obrascem, prikaza obrasca te statusne trake na dnu prozora. Nakon inicijalizacije predloška korisnik može definirati pozicije oznaka čime definira koordinatni sustav obrasca te može definirati pozicije ćelija koje sadrže bodove studenata pritiskom tipke miša unutar ćelija tablice. Na slici 24 prikazano je stanje prikaza nakon završene definicije predloška obrasca.



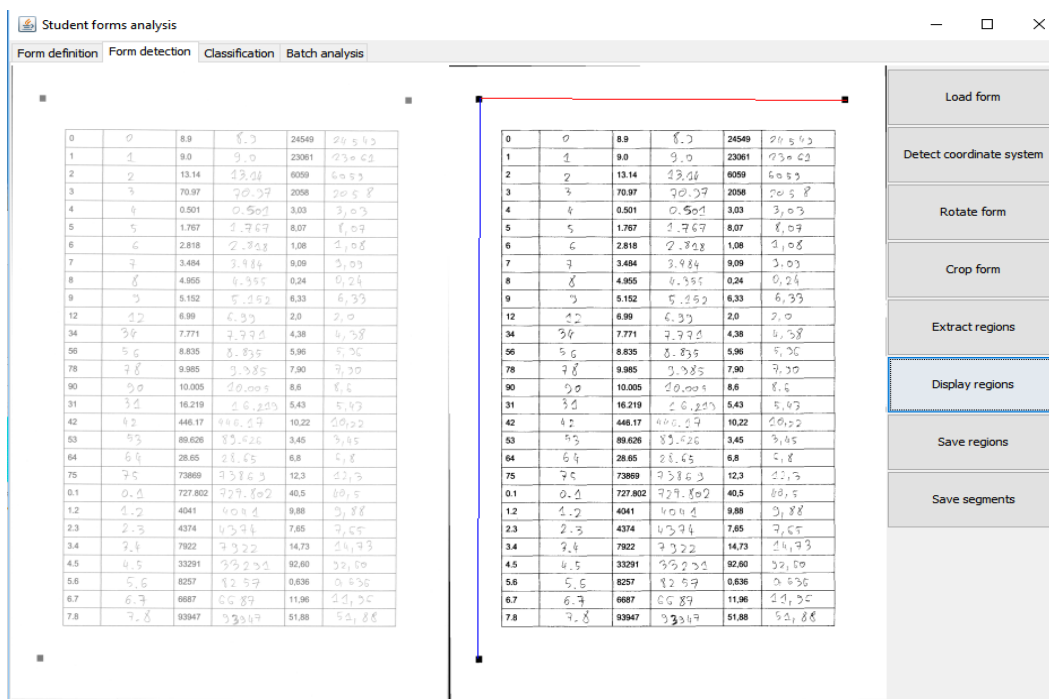
Slika 23. Prikaz praznog prikaza za definiciju predloška obrasca



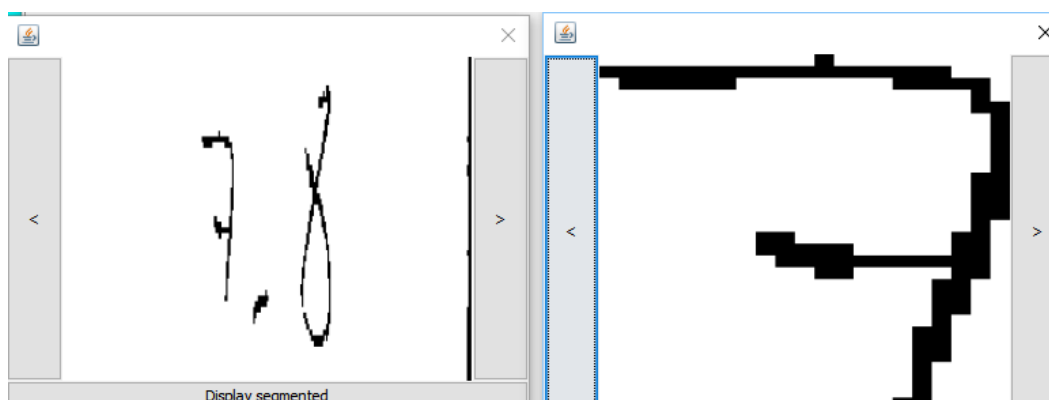
Slika 24. Prikaz prozora za definiciju predloška obrasca s definiranim obrascem

- Prikaz za analizu formulara. Prikaz omogućava korisniku da postupno transformira skenirani obrazac, detektira koordinatni sustav temeljem definiranog predloška obrasca te da izdvoji ćelije s bodovima studenata. Korisnik može pregledati pojedine ćelije te segmente na koje se segmentira broj unutar ćelije.

Slika 25 prikazuje prikaz za analizu formulara s transformiranim obrascem, a slika 26 prikazuje ćeliju obrasca te segment unutar ćelije.



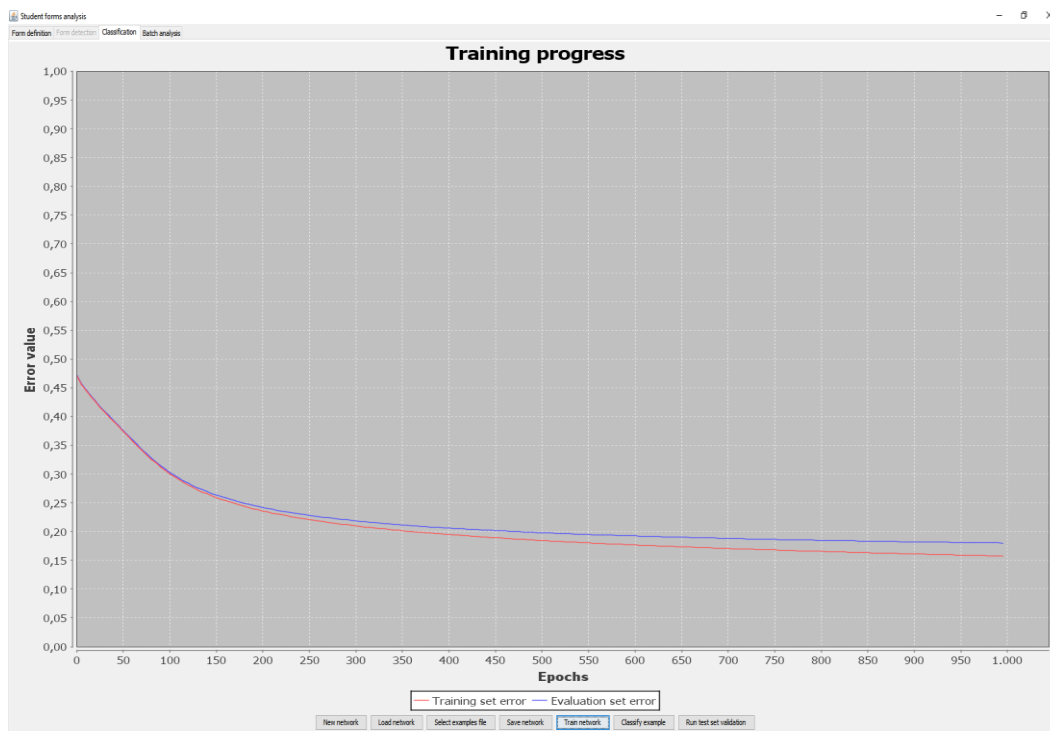
Slika 25. Prikaz za analizu obrasca s transformiranim obrascem



Slika 26. Prozor s ćelijom s bodovima studenta te prozor sa segmentom broja

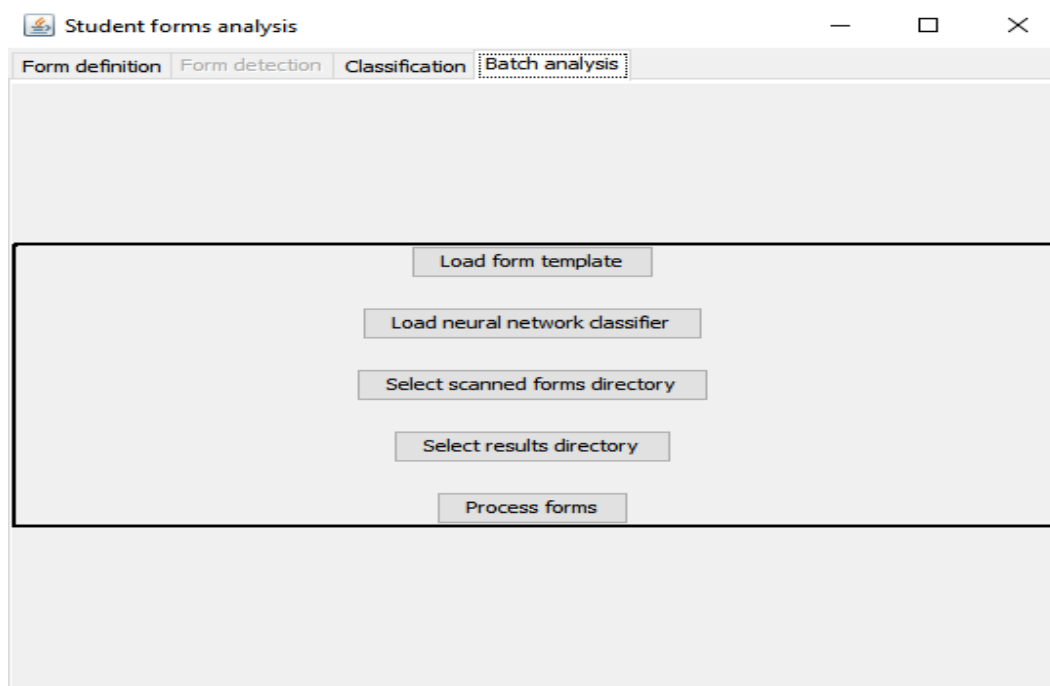
- Prikaz za treniranje neuronske mreže. Prikaz omogućava korisniku treniranje neuronske mreže za klasifikaciju znamenki. Korisnik može definirati neuronsku mrežu, skup uzoraka za učenje te parametre treniranja. Prilikom treniranja prikaz prikazuje graf ovisnosti greške na skupu za treniranje i skupu za provjeru u pojedinoj epohi treniranja. Korisnik ujedno može klasificirati pojedini primjer.

Slika 27 prikazuje prikaz za treniranje neuronske mreže.



Slika 27. Prikaz za treniranje neuronske mreže

- Prikaz za obradu skeniranih obrazaca. Prikaz omogućava korisniku obradu skupa skeniranih obrazaca. Korisnik treba učitati predložak obrasca ako već nije učitao u program, definiranu neuronsku mrežu, direktorij u kojem se nalaze skenirani obrasci te direktorij u koji je potrebno spremiti rezultate. Slika 28 prikazuje prikaz za obradu skeniranih obrazaca.



Slika 28. Prikaz za obradu skeniranih obrazaca

## 6.2. Primjer izlazne datoteke

Rezultat obrade obrazaca je niz izlaznih datoteka formatiranih u TSV formatu (engl. *tab-separated values*) u kojem su vrijednosti odvojene tabulatorom. Datoteka se sastoji od retka zaglavlja koje sadrži polja *redni broj*, *broj bodova* i *poruka greške* te redaka vrijednosti polja zaglavlja za svaku ćeliju unutar obrasca. Polje *redni broj* poprima vrijednosti od jedan do broja ćelija unutar obrasca, polje *broj bodova* poprima vrijednosti koje su prepoznate unutar pojedine ćelije, a polje *poruka greške* sadrži moguću poruku sustava o sumnji na pogrešku prilikom procesa obrade obrasca. Podržane prepoznate moguće pogreške su situacije u kojima sustav prepozna više od jednog decimalnog separatora te situacija u kojoj postoji vodeća nula iza koje ne slijedi decimalni separator.

Primjer izvotka iz izlazne datoteke dan je na slici 29. Na slici je dan dio datoteke koji prikazuje ispravnu obradu ćelija, neispravnu obradu ćelija te neispravnu obradu ćelija s porukom o prepoznatoj pogrešci.

1	Field number	Number of points	Error message
2		0	0
3		1	1
4		2	2
5		3	3
6		4	4
7		5	5
8	-----		
9		35	9.484
10		36	4.955
11		37	4..52
12		38	6.99
13		39	7771
14	-----		
15			
16		65	0733
17		66	40
			WARNING: Possible missclassification leading zero detected

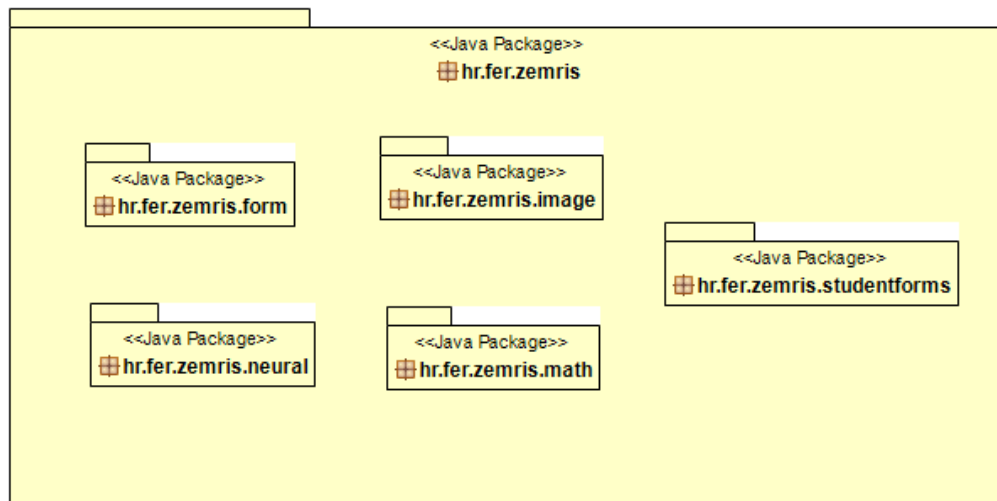
Slika 29. Primjer djela izlazne datoteke

## 6.3. Raspored komponenata unutar programskih paketa

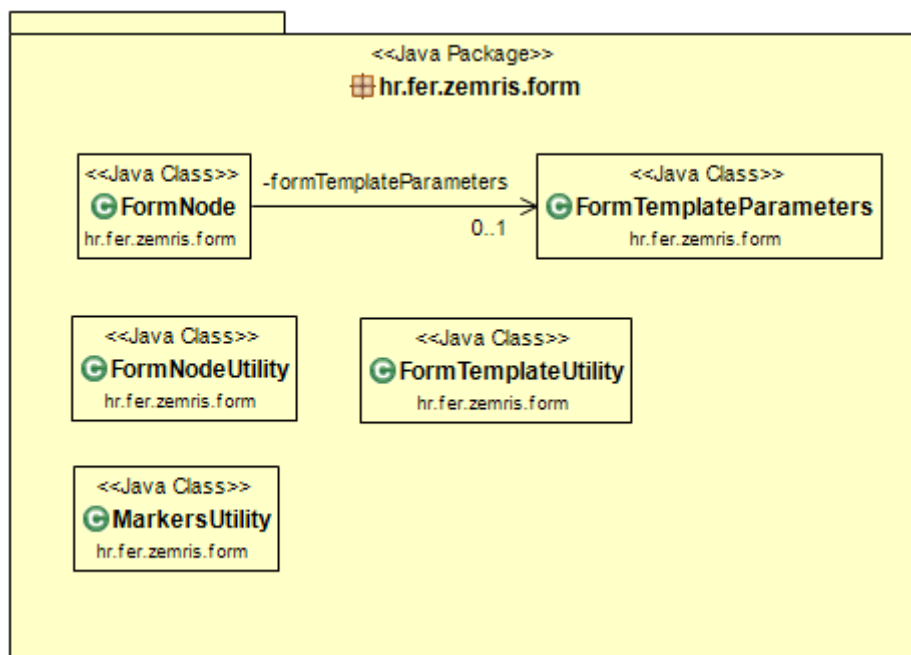
U okviru ovog potpoglavlja dan je pregled implementacije programskog sustava prema programskim paketima. Opći raspored paketa dan je na slici 30. Programski kod je raspoređen u pet paketa: `form`, `image`, `neural`, `math` i `studentforms`.

Paket `form` sadrži razrede zadužene za reprezentaciju formulara (`FormNode`), predloška formulara (`FormTemplateParameters`) te razrede zadužene za specifičnu obradu

obrazaca (FormNodeUtility, FormTemplateUtility, MarkersUtility).  
 Detaljna struktura paketa form dana je na slici 31.



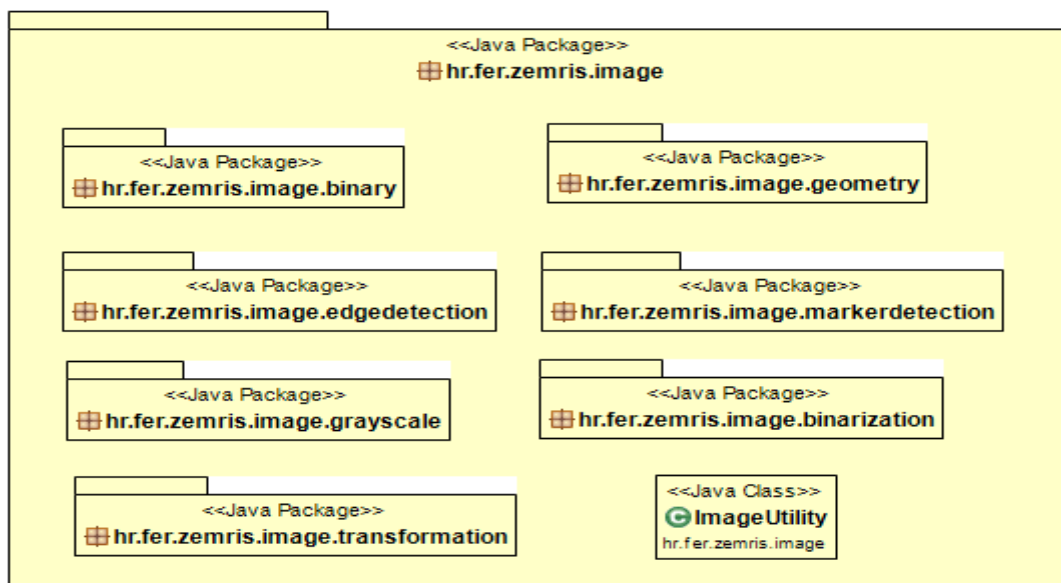
Slika 30. Opći raspored paketa



Slika 31. Prikaz paketa form

Paket image sadrži: paket zadužen za reprezentaciju binarne slike (binary), pakete zadužene za algoritme pretvaranja boje u nijanse sive (grayscale) i algoritme binarizacije (binarization), paket za detekciju oznaka na slici (markerdetection), paket za detekciju rubova (edgedetection), paket za prikaz i obradu geometrijskih oblika na slikama (geometry), paket za transformacije nad slikama te pomoćni razred za opće operacije nad slikama (ImageUtility). Detaljan prikaz paketa image dan je na slici 32.



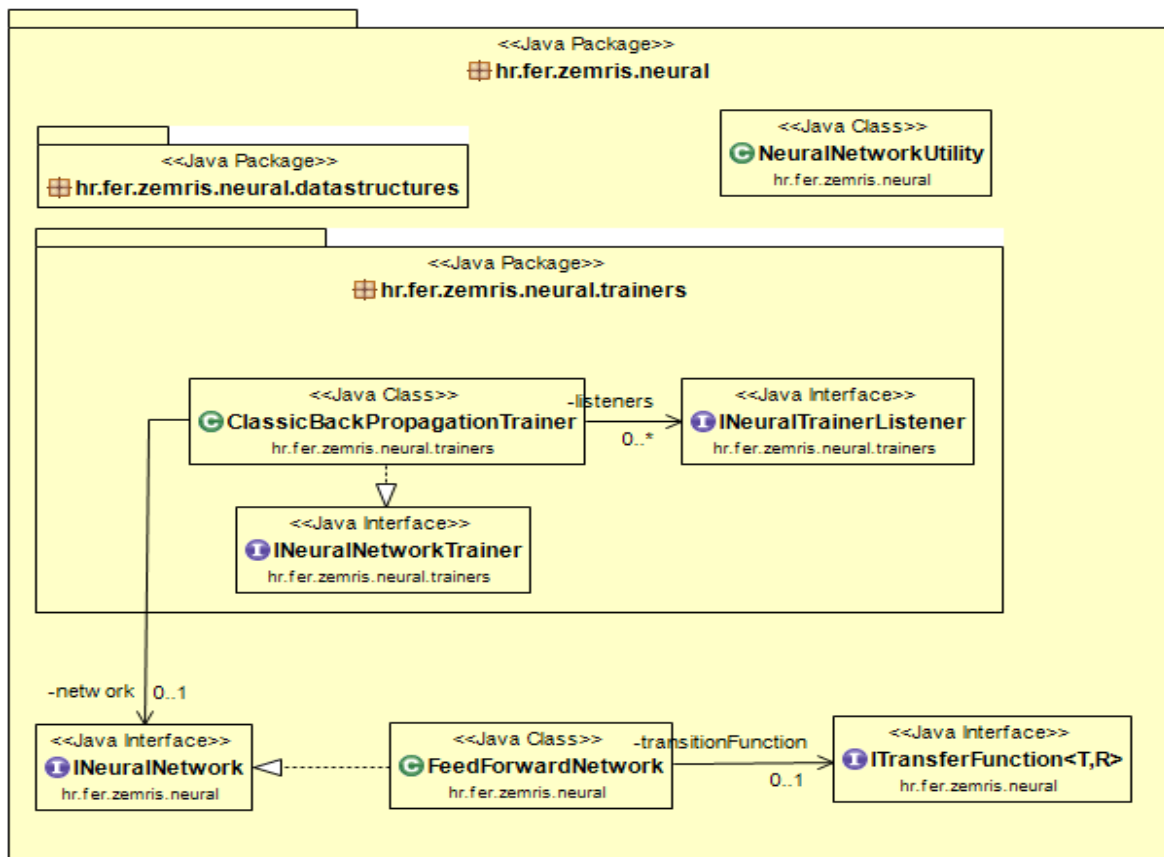


Slika 32. Prikaz paketa image

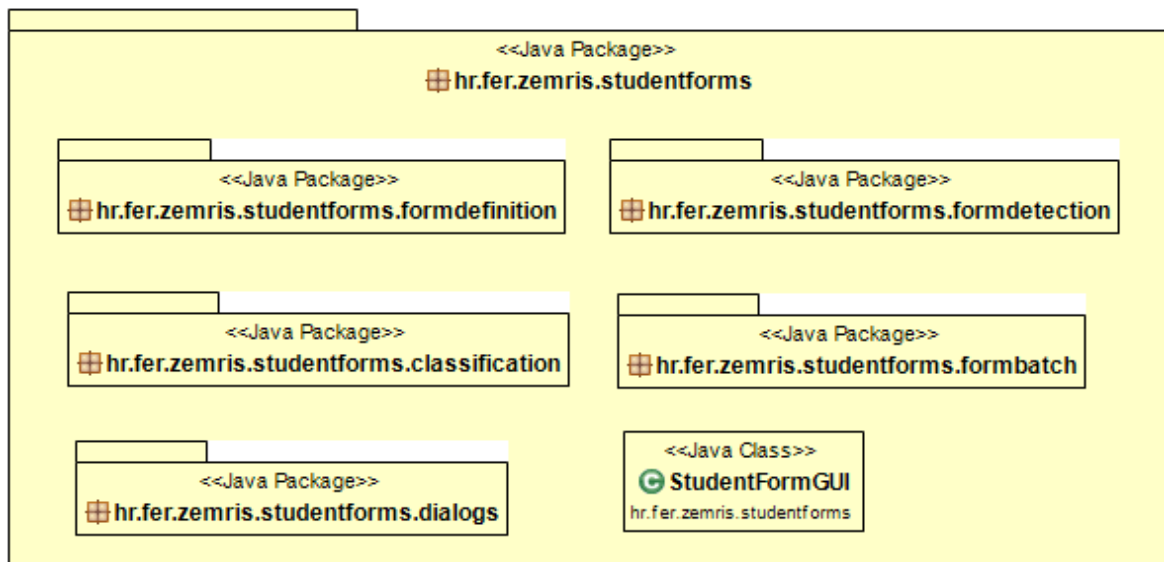
Paket `neural` sadrži sučelja i razrede vezane uz reprezentaciju neuronske mreže (`INeuralNetwork`, `FeedForwardNetwork`), reprezentaciju trenera neuronske mreže (`INeuralNetworkTrainer`), implementaciju sustava za treniranje neuronske mreže algoritmom propagacije pogreške unatrag (`ClassicBackPropagationTrainer`), sučelje za promatranje napretka treniranja (`INeuralTrainerListener`), reprezentaciju prijenosne funkcije (`ITransferFunction`), razred za pomoćne operacije s neuronskim mrežama (`NeuralNetworkUtility`) te paket koji sadrži strukture za pohranu ulaznih primjera za treniranje neuronske mreže (`datastructures`). Detaljan prikaz paketa `neural` dan je na slici 33.

Paket `math` sadrži razred s operacijama za statističku analizu niza brojeva.

Paket `studentforms` sadrži razred koji predstavlja glavni prozor programa (`StudentFormGUI`), pakete za pojedine prikaze i paket s posebnim dijalozima (`dialogs`). Paket `formdefinition` sadrži razrede korištene u prikazu za definiciju predloška obrasca, paket `formdetection` sadrži razrede korištene u prikazu za analizu pojedinog obrasca, paket `classification` sadrži razrede korištene u prikazu za treniranje neuronske mreže, paket `formbatch` sadrži razrede korištene u prikazu za obradu skeniranih obrazaca. Poseban dijalog definiran u paketu `dialogs` korišten je za prikaz niza slika u jednom zasebnom prozoru. Detaljan prikaz paketa `studentforms` dan je na slici 34.



Slika 33. Prikaz paketa neural



Slika 34. Prikaz paketa studentforms

## 7. Rezultati obrade obrazaca

U okviru ovog završnog rada ulazni skup podataka se sastojao od 22 skenirana obrasca. Svaki skenirani obrazac je imao 84 ćelije s bodovima studenata. Pojedini broj bodova studenata je imao između jednog i sedam znakova. Pri brojanju znakova brojali smo broj znamenki i moguće decimalne separatore.

Proces obrade skeniranog obrasca prikazan je na slici 2 na 3. stranici te je opisan u poglavlju 2.

Postupak segmentacije je bio uspješan u 78.1% polja bodova studenata kao što je detaljno prikazano u poglavlju 4.3. Bitno je napomenuti da je uspješnost segmentacije mjerena prema broju izdvojenih segmenata.

Postupak klasifikacije je bio uspješan u 82.41% ulaznih primjera znakova za testiranje. Na cjelokupnom skupu znakova koji su uspješno segmentirani uspješnost je iznosila 87%.

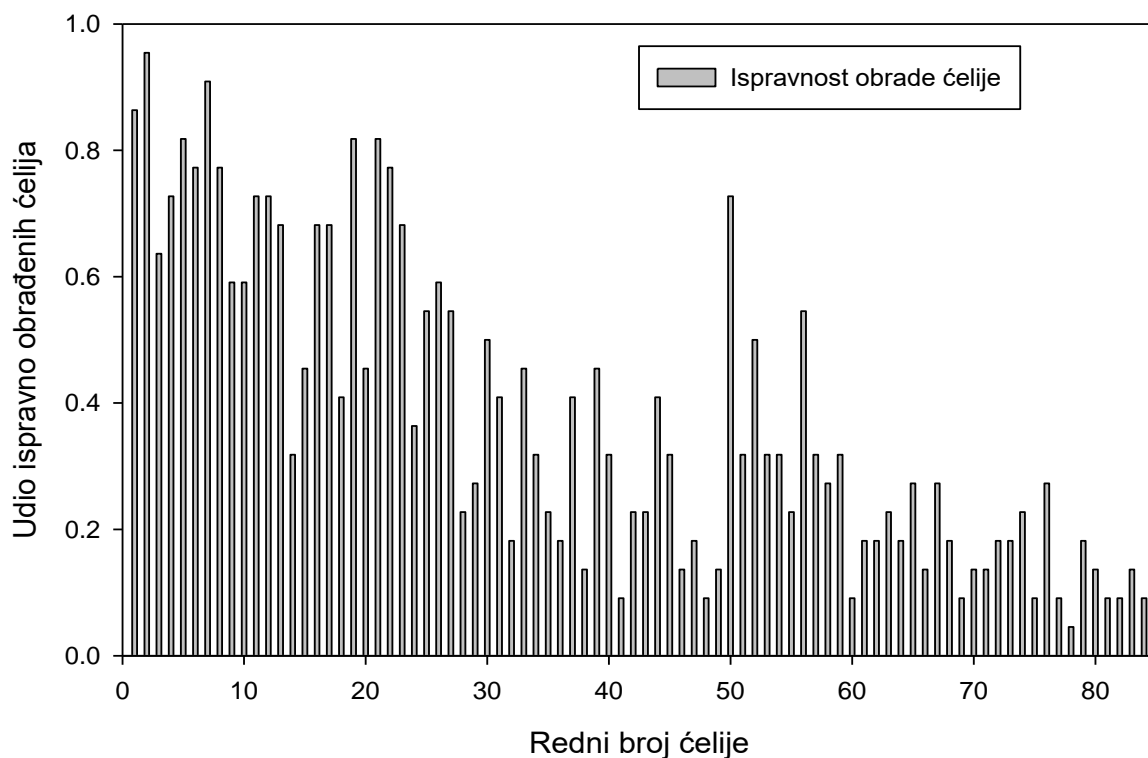
Ukupan rezultat obrade obrazaca po ćelijama s bodovima studenata iznosi 38% od ukupno 1848 ćelija. Maksimalni iznos ispravne obrade ćelije je dobiven za ćeliju koja je sadržavala broj „1“ i iznosio je 95%, a minimalni iznos ispravne obrade ćelije je dobiven za ćelije čije su vrijednosti {8.835, 73869, 2058, 5.96, 40.5, 92.6, 0.636, 51.88} i iznosio je 9.1%.

Razmotrimo ukratko jednu od vrijednosti koja je imala minimalni iznos ispravne obrade kako bi ukazali na razloge zašto je vrijednost loše obrađena. Za primjer smo odabrali broj „73869“. Broj je uspješno segmentiran po broju segmenata u 77.27% primjera. A znamenke od koji se sastoji su uspješno klasificirane u sljedećim stopama uspješnosti: 86%, 81%, 58%, 0.99%, 0.8%. Očekivanu vjerojatnost uspješne obrade možemo računati sljedećim postupkom.

$$p = 0.7727 \cdot (0.86 \cdot 0.81 \cdot 0.57 \cdot 0.99 \cdot 0.8) = 0.243$$

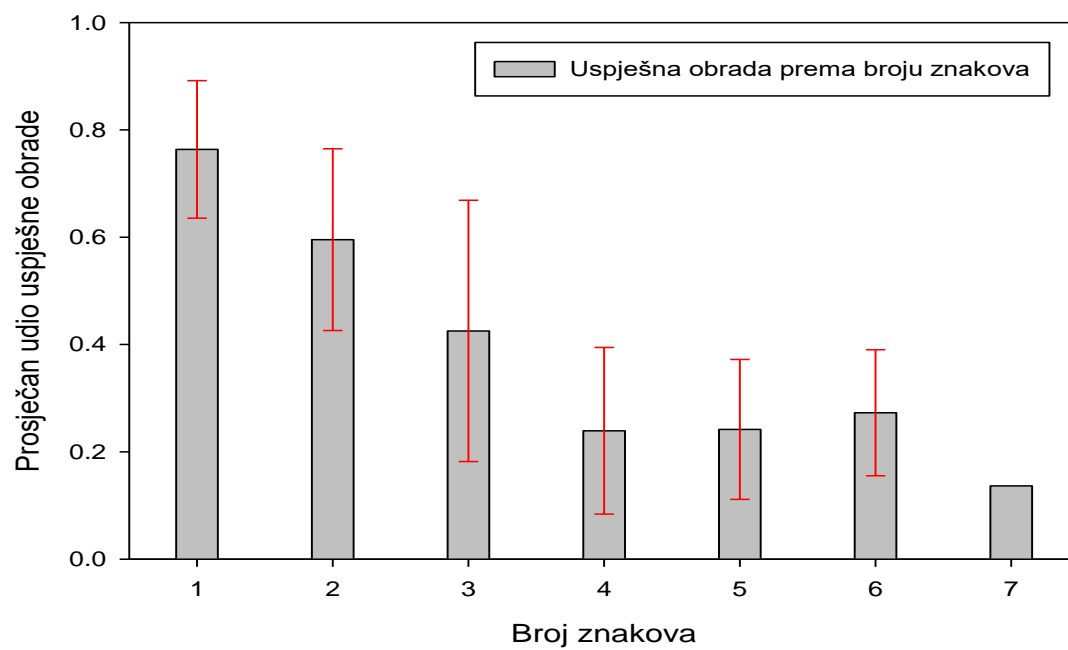
Bitno je napomenuti kako je na izračun vjerojatnosti utjecalo to što je broj uspješnosti segmentacije mjeran prema broju uspješno segmentiranih segmenata, a prava stopa uspješnosti segmentacije je manja ili jednaka navedenoj. Možemo primijetiti kako pojavljivanje broja „8“ jako utječe na vjerojatnost uspješne obrade zbog toga što ta znamenka ima minimalnu stopu uspješne klasifikacije koju daje korišteni klasifikator po pojedinim znamenkama. Drugi razlog zašto je broj loše obrađen je i broj znamenaka koje broj ima jer se svakom znamenkom trenutna vjerojatnost množi s brojem manjim od 1.

Na grafu 8 dan je detaljan prikaz uspješnosti obrade ćelija prema broju ćelije, a vrijednosti bodova su korištene prema predlošku iz obrasca danog na slici 1 na 2. stranici. Ćelije su numerirane tako što su označene rednim brojem od 1 do 84 pri čemu su prvo brojevi dodijeljeni prvom stupcu, zatim drugom te naposljetku trećem stupcu.



Graf 8. Udio ispravno obrađenih ćelija prema vrijednosti ćelije

Na grafu 9 prikazan je prosječan udio uspješne obrade ćelija prema broju znakova. Crvenom bojom prikazano je standardno odstupanje uspješnosti obrade ćelije s određenim brojem znakova. Na temelju grafa se može zaključiti kako uspješnost obrade ćelije pada s brojem znakova.



Graf 9. Prosječan udio uspješne obrade ćelija prema broju znakova (brojane su znamenke i decimalni separatori)

## 8. Zaključak

Prilikom ispravljanja ispita koje piše veliki broj studenata ukazuje se potreba za sustavom koji može bodove unesene u obrazac za ocjenjivanje unijeti u računalni sustav tako što provede lokalizaciju i očitavanja rukom pisanih bodova studenata.

U okviru ovog završnog rada opisan je i implementiran sustav za lokalizaciju i očitavanja rukom pisanih bodova studenata. Proces obrade obrasca za ocjenjivanje se sastoji od skeniranja obrasca, obrade slike, segmentacije znamenki bodova, klasifikacije znamenki te kreiranja izlazne datoteke.

Obrada slike je provedena zbog smanjenja nepotrebnih informacija sadržanih u skeniranom obrascu te kako bi se iz skeniranog obrasca izrezale ćelije s bodovima studenata. Prezentirano je nekoliko algoritama pretvaranja slike u nijanse sive te binarizacije slike. Za obradu obrasca odabrani su algoritam minimalnog intenziteta za pretvaranje boje u nijanse sive te globalni algoritam fiksnog praga uz prag jednak 250 za binarizaciju slike.

Segmentacija slike je provedena koristeći algoritam povezanih komponenti nakon kojeg je provedeno filtriranje suvišnih segmenata. Korišteni postupak segmentacije je postizao ukupnu stopu uspješne segmentacije od 78.51%, a ukupna stopa prekomjerne segmentacije je iznosila 6.27% te ukupna stopa nedovoljne segmentacije 15.22% primjera. Uspješnost je mjerena prema broju izdvojenih segmenata.

Klasifikacija znamenaka je provedena koristeći unaprijednu slojevitú neuronsku mrežu treniranu algoritmom propagacije pogreške unatrag. Korišteni klasifikator je postizao 82.41% uspješne stope klasifikacije na skupu primjera znakova za testiranje. Na cjelokupnom skupu znakova koji su uspješno segmentirani uspješnost je iznosila 87%.

Rezultat ukupne obrade obrazaca je iznosio 38% uspješne obrade ćelija od ukupno 1848 obrađenih ćelija koje sadrže bodove studenata.

Nastavak razvoja i istraživanja sustava bi prvenstveno trebao biti u smjeru kvalitetnije segmentacije znamenaka kako bi se izbjegla pretjerana segmentiranost te kako bi se prepoznale spojene znamenke za koje bi trebalo generirati hipoteze segmentacije te zatim odabrati najbolju hipotezu. Kvaliteta segmentacije u trenutnom sustavu predstavlja prepreku za povećanje stope uspješne obrade skeniranih obrazaca.

# Literatura

1. Čupić, Marko; Hrkać, Tomislav; Mihajlović, Željka; Kalafatić, Zoran: „Automatic recognition of handwritten corrections for multiple-choice exam answer sheets“, MIPRO 2014, Opatija, Hrvatska, 2014.
2. Suratane, Apichat; Lertsari, Nantaporn; Kamphasee, Sethawat; Sriket, Kritsada: „Handwritten digit recognition for managing examination score in paper-based test“, „Journal of Theoretical and Applied Information Technology“, vol. 65., no. 3., pp. 633-638, srpanj 2014.
3. Ferilli, Stefano: „Automatic Digital Document Processing and Management“, Springer, New York, USA, 2011.
4. Puneet, Naresh Kumar Garg: "Binarization Techniques used for Grey Scale Images", International Journal of Computer Applications, Vol. 71, No. 1, pp. 8-11, June 2013.
5. Otsu, Nobuyuki: „A Threshold Selection Method from Gray-Level Histograms“, IEEE Transactions on Systems, Man, and Cybernetics, vol. 9., no. 1, pp. 62-66, siječanj 1979.
6. Casey, Richard G.; Lecolinet, Eric: „A Survey of Methods and Strategies in Character Segmentation“, IEE Transaction on Pattern Analysis and Machine Intelligence, vol. 18, no. 7, pp. 690-706, srpanj 1996.
7. Ribas, F. C.; Oliveira, L. S.; Britto Jr., A. S.; Sabourin, R.: „Handwritten digit segmentation: a comparative study“, International Journal on Document Analysis and Recognition, vol. 16, no. 2, pp. 127-137, lipanj 2013.
8. Jain, Ramesh; Kasturi, Rangachar; Schunck, Brian G.: „Machine Vision“, McGraw-Hill, Inc., pp. 28.-38., New York, USA, 1995.
9. Jankowski, Mariusz; Kuska, Jens-Peer: „Connected components labeling – algorithms in Mathematica, Java, C++ and C#“, International Mathematica Symposium, Banff, Canada, 2004.
10. Stockman, George C.; Shapiro, Linda G.: „Computer Vision“, Pearson, 1. izdanje, pp. 69-75, veljača 2001.
11. Sauvola, J.; Pietikäinen, M.: „Adaptive document image binarization“, The Journal of the Pattern Recognition Society, vol 33. no. 2, pp. 225-236, veljača 2000.

12. Khurshid, Khurram; Siddiqi, Imran; Faure, Claudie; Vincent, Nicole: „Comparison of Niblack inspired binarization methods for ancient documents“, Document Recognition and Retrieval Conference XVI, San Jose, CA, USA, siječanj 2009.
13. Čupić, Marko; Dalbelo Bašić, Bojana; Glub, Marin: „Neizrazito, evolucijsko i neuroračunarstvo“, Fakultet elektrotehnike i računarstva, Sveučilište u Zagrebu, 12. kolovoza 2013.
14. Čupić, Marko: „Umjetna inteligencija – Umjetne neuronske mreže“, javno dostupno na web stranici <http://java.zemris.fer.hr/nastava/ui/> , prvo izdanje, svibanj 2016.
15. Fujisawa, H.; Nakano, Y.; Kurino, K.: „Segmentation methods for character recognition: from segmentation to document structure analysis“, Proceedings of IEEE, vol. 80, no. 7, pp. 1079-1092, srpanj 1992.
16. Čupić, Marko; Mihajlović, Željka: „Interaktivna računalna grafika kroz primjere u OpenGL-u“, Fakultet elektrotehnike i računarstva, Sveučilište u Zagrebu, pp 101-104, 23. rujna 2016.
17. „Flood Fill“, Wikipedia, mrežna stranica: [https://en.wikipedia.org/wiki/Flood\\_fill](https://en.wikipedia.org/wiki/Flood_fill) , zadnje pristupljeno 31. svibanj 2017.
18. LeCun, Yann; Cortes, Corinna; Burges, Christopher J.C. : „The MNIST database of handwritten digits“, mrežna stranica: <http://yann.lecun.com/exdb/mnist/>, zadnje pristupljeno 3. lipanj 2017.
19. Lei He, Chun; Zhang, Ping; Dong, Jianxiong; Suen, Ching Y.; Bui, Tien D. : „The Role of Size Normalization on the Recognition Rate of Handwritten Numerals“, Neural Networks and Learning in Document Analysis and Recognition, Seoul, Korea, kolovoz 2005.
20. Bunke, Horst; Wang, Patrick Shen-pei : „Handbook of Character Recognition and Document Image Analysis“, Word Scientific, pp. 325, 1997.
21. Zhao, Bin; Su, Hui; Xia, Shaowei: „A New Method for Segmenting Uncronstrained Handwritten Numeral String“, Proceedings of the Fourth Conference on Document Analysis and Recognition, pp 524-527, kolovoz 1997.
22. Michael Nielsen: „Using neural nets to recognize handwritten digits“, mrežna stranica: <http://neuralnetworksanddeeplearning.com/chap1.html>, zadnje pristupljeno 3. lipanj 2017.



# Sažetak

## Lokalizacija i očitavanja rukom pisanih bodova studenata

Prilikom ispravljanja ispita koje piše veliki broj studenata ukazuje se potreba za sustavom za raspoznavanje rukom pisanih bodova studenata sa skeniranih slika.

U okviru ovog završnog rada opisan je i implementiran sustav za lokalizaciju i očitavanja rukom pisanih bodova studenata. Proces obrade obrasca za ocjenjivanje se sastojao od skeniranja obrasca, obrade slike, segmentacije znamenki bodova, klasifikacije znamenki te stvaranja izlazne datoteke.

Za obradu obrasca odabrani su algoritam minimalnog intenziteta za pretvaranje boje u nijanse sive te globalni algoritam fiksnog praga za binarizaciju slike. Segmentacija slike je provedena koristeći algoritam povezanih komponenti nakon kojeg je provedeno filtriranje suvišnih segmenata. Korišteni postupak segmentacije je postizao ukupnu stopu uspješne segmentacije od 78.51%. Klasifikacija znamenaka je provedena koristeći unaprijednu slojevitú neuronsku mrežu treniranu algoritmom propagacije pogreške unatrag. Korišteni klasifikator je postizao 82.41% uspješne stope klasifikacije na skupu primjera znamenki za testiranje.

Rezultat ukupne obrade obrazaca je iznosio 38% uspješne obrade ćelija od ukupno 1848 obrađenih ćelija.

**Ključne riječi:** raspoznavanje rukom pisanih znamenaka, segmentacija znamenaka, klasifikacija znamenaka, neuronska mreža, propagacija pogreške unazad, lokalizacija teksta, obrada slike, algoritam označavanja povezanih komponenti

# Summary

## Localization and Recognition of Handwritten Student Scores

When examining a large number of student exams, there is a need for an automatic system for localization and recognition of handwritten student scores.

In this bachelor's thesis a system for localization and recognition of handwritten student scores has been analysed and implemented. The analysis process consisted of form scanning, image processing, digit segmentation, digit classification and output file creation.

For processing the form, a minimal intensity algorithm for converting image to grayscale was used and a fixed global threshold algorithm for image binarization. Number segmentation was performed using the connected-component labeling algorithm after which the filtering of the redundant segments was performed. The segmentation process used achieved the overall rate of successful segmentation of 78.51%. Digit classification was performed using feedforward neural network trained with backpropagation algorithm. The classifier used achieved successful classification rate of 82.41% on test set. The total result of the form processing system was 38% of the total of 1848 processed form fields.

**Keywords:** handwritten digit recognition, digit segmentation, digit classification, feedforward neural network, backpropagation, text localization, image processing, ocr, connected-component labeling algorithm