SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 180

RASPOZNAVANJE ZNAKOVA NA REGISTARSKIM TABLICAMA

Anđelo Martinović

Zagreb, lipanj 2008.

Sadržaj

1. Uvod 2

1.1. Programski sustav 2
1.2. Sklopovska podrška 3
2. Lokalizacija registarske tablice 5

2.1. Otkrivanje rubova 5
2.1.1. Otkrivanje horizontalnih rubova 6
2.1.2. Otkrivanje vertikalnih rubova 6
2.2. Horizontalna i vertikalna projekcija 7
2.3. Lokalizacija tablice 8
2.3.1. Izdvajanje trake 8
2.3.2. Izdvajanje tablice 11
3. Segmentacija tablice 13

3.1. Binarizacija slike 13
3.2. Segmentacija pomoću horizontalne projekcije 14
3.3. Izdvajanje znakova iz segmenata 15
3.3.1. Traženje povezanih komponenti 15
3.3.2. Heuristička analiza povezanih komponenti 17
4. Analiza znakova 18

4.1. Normalizacija veličine 18
4.2. Izvlačenje značajki 19
4.2.1. Algoritam skeletonizacije 19
4.2.2. Strukturalna analiza 22
5. Klasifikacija znakova 25

5.1. Opći problem klasifikacije 25
5.2. Pozicijska analiza 27
5.3. Neuronske mreže 28
5.3.1. Biološki neuron 28
5.3.2. Matematički modeli neurona 29
5.3.3. Višeslojna neuronska mreža s unaprijednom propagacijom 30
5.3.4. Primjena neuronske mreže na klasifikaciju znakova 31
6. Sintaksna analiza 34

6.1. Sintaksni uzorci 34
6.2. Primjena uzoraka 35
7. Testovi i završna razmatranja 36

7.1. Baza slika 36
7.2. Uspješnost prepoznavanja 37
8. Zaključak 38

9. Literatura 39

Dodatak A: Programska potpora 41

1. Uvod

Prvi sustav za automatsko raspoznavanje registarskih tablica (engl. *Automatic Number Plate Recognition – ANPR*) je izumljen 1976. godine u Ujedinjenom Kraljevstvu. Rani sustavi su bili većinom nepouzdani, s niskim stopama prepoznavanja. Tijekom proteklih trideset godina brzi razvoj računalno-informacijske infrastrukture omogućio je stvaranje sve preciznijih i pouzdanijih sustava. Danas, ANPR tehnologija omogućava stope prepoznavanja i do 98%. Neki sustavi podržavaju prepoznavanje registarskih tablica na vozilima koja se kreću i do 200 km/h. [1]

Primjena ANPR sustava je raznolika. Na parkiralištima, sustav se može koristiti za bilježenje podataka o vozilu i automatsku naplatu ovisno o vremenu parkiranja. Također, ANPR sustavi se koriste i za kontrolu pristupa na privatnim posjedima, čime se povećava sigurnost tvrtke i pojedinaca. Državne službe koje se bave kontroliranjem prometa iskorištavaju ANPR sustave kako bi pomoću distribuirane mreže kamera pronašli točno određeno vozilo, odredili prosječne brzine vožnje na prometnicama ili pak za optimizaciju prometa preusmjeravanjem vozila s preopterećenih ruta na zamjenske.

1.1. Programski sustav

Jedno od temeljnih pitanja u problematici automatskog prepoznavanja registarskih tablica jest kako računalu predočiti tablicu. Čovjeku je to jednostavan posao, dok stroju registarska tablica ne znači ništa više od skupa piksela. Formalno, računalu se predaje jednostavna dvodimenzionalna funkcija $f(x, y)$, gdje su x i y prostorne koordinate pojedine točke na slici, a f je intenzitet svjetla u toj točki. Kako bi računalo moglo „pročitati“ registarsku tablicu potrebno je dizajnirati kompleksni matematički sustav koji može izvući semantiku iz zadanog ulaza na temelju njegovih karakterističnih značajki.

Definicija registarske tablice izrečena jednostavnim jezikom bi glasila otprilike: „Mala plastična ili metalna ploča pričvršćena za vozilo koja služi za njegovu identifikaciju“. Računala ne razumiju ovu definiciju, te se mora pronaći neki drugi način za formuliranje uvjeta po kojima će računalo ispravno prepoznati mjesto na slici gdje se nalazi registarska tablica. Poglavlje 2 se bavi upravo ovom problematikom, te se opisuju matematički algoritmi za brzu i jednostavnu detekciju područja registarske tablice, s naglaskom na invarijantnost sustava o promjeni boje, osvjetljenja ili kuta gledanja.

Nakon što je registarska tablica pronađena potrebno je izvršiti segmentaciju dobivenog područja. Algoritam bi trebao biti u mogućnosti ispravno odvojiti znakove, odbacujući neispravne i prosljeđujući pronađene segmente na daljnju obradu. Navedeni algoritam je opisan u poglavlju 3.

Dijelovi registarske tablice dobiveni nakon segmentacije često nisu prikladni za računalnu obradu. Prije klasifikacije, potrebno je izlučiti karakteristične značajke pojedinog segmenta. One predstavljaju kompaktnu i apstraktnu reprezentaciju segmenta i kao takve su prikladne za daljnje obrađivanje. Poglavlje 4 sadrži opis izlučivanja reprezentativnih vektora.

Poglavlje 5 se bavi klasifikacijom znakova. Svi dozvoljeni znakovi na registarskim tablicama se dijele u klase ovisno o karakterističnim značajkama. Za neke klase znakova je dovoljan jednostavan algoritam pozicijske analize, dok se za većinu znakova koristi klasifikator u vidu neuronske mreže. Opisana su teorijske osnove neuronskih mreža i njihova primjena u sustavu za raspoznavanje znakova.

U poglavlju 6 je prikazan zadnji korak raspoznavanja: sintaktička analiza. Svaka država ima svoj skup sintaktičkih pravila za definiranje registarskih tablica. Ukoliko nakon provedene analize nije dobiven ispravan rezultat, ta informacija se može iskoristiti za ispravljanje pogreške nastale u nekom od prijašnjih koraka procesa prepoznavanja. Ukoliko se od sustava zahtijeva da prepoznaje registarske tablice više država, nadogradnja sustava se često može svesti na definiranje nekoliko jednostavnih sintaktičkih pravila.

1.2. Sklopovska podrška

Slika koja se obrađuje u ANPR sustavu najčešće dolazi iz specijaliziranih kamera. Faktori koji najviše utječu na kvalitetu prepoznavanja uključuju brzinu vozila koje se snima, varirajuće uvjete osvjetljenja, blještanje prednjih svjetala i oštre vremenske uvjete. Neke kamere koriste infracrveni dio spektra kako bi zaobišle problem osvjetljenja i reflektivnosti registarske tablice.

Mnoge zemlje danas koriste retroreflektivne registarske tablice. [2] Takve ploče reflektiraju svjetlost natrag prema izvoru čime poboljšavaju kontrast slike. U nekim zemljama samo znakovi na tablici nisu reflektivni, čime se postiže visok stupanj kontrasta prema reflektivnoj podlozi. Infracrvene kamere s pripadajućim izvorom infracrvene

svjetlosti imaju naročito veliku korist od ovog principa, pošto se infracrveni valovi reflektiraju od tablice.

Zamagljene slike otežavaju postupak prepoznavanja, a u nekim slučajevima ga i potpuno onemogućuju. ANPR sustavi moraju imati jako malo vrijeme ekspozicije (i do 1/1000 s) kako bi se izbjeglo zamagljenje zbog pokreta (engl. *motion blur*).

Relativno pozicioniranje kamere u odnosu na vozila je također jako bitan faktor. Manje visine omogućuju direktniji pogled na tablicu, no povećava se vjerojatnost prekrivanja tablice drugim objektom. Nasuprot tomu, kamera na povišenoj poziciji u većini slučajeva ima nezaklonjen pogled na tablicu, no povećava se kut pod kojim se tablica vidi na dobivenoj slici, čime dolazi do distorzije i iskošenja slike.

2. Lokalizacija registarske tablice

Prvi i temeljni problem koji se nameće na samom početku procesa prepoznavanja jest ispravno definiranje pojma “registarska tablica”. Standardna definicija “male metalne identifikacijske pločice” u svijetu računala nije primjerena. Zato ćemo za potrebe algoritma detekcije preformulirati početnu definiciju, i reći da je registarska tablica *pravokutno područje na slici s povećanom koncentracijom horizontalnih i vertikalnih rubova*. Iako će u većini slučajeva velika razlika u kontrastu između slova i pozadine tablice omogućiti ispravno funkcioniranje algoritma koji se zasniva na ovoj definiciji, postojat će i slučajevi u kojima će zahtjevna pozadina uzrokovati nepoželjno funkcioniranje algoritma. Kako bi zaobišao ovaj problem, algoritam prepoznavanja će odrediti više mogućih kandidata za tablicu, te će svako područje biti prihvaćeno ili odbačeno na temelju daljnje heurističke analize.

Prvi korak u lokalizaciji registarske tablice jest određivanje rubova na slici. Ovaj algoritam se zasniva na nizu operacija konvolucije nad početnom slikom. Kasnije se dobiveni rezultati projiciraju na koordinatne osi.

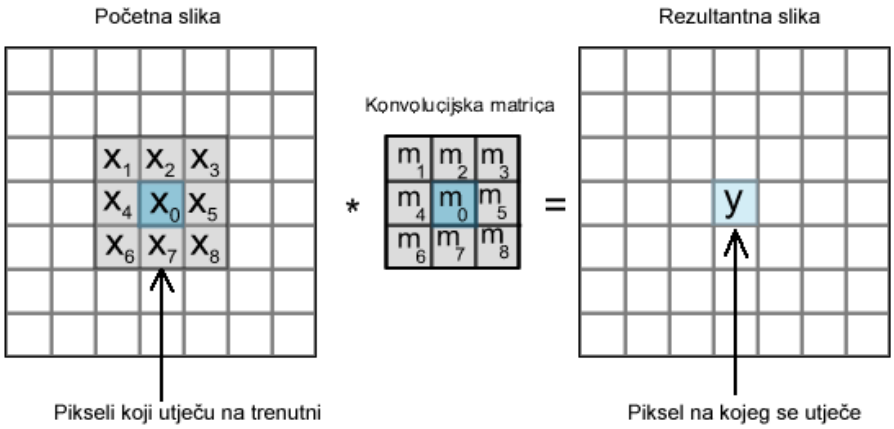
2.1. Otkrivanje rubova

Najjednostavniji način za otkrivanje različitih vrsta rubova na slici jest konvoluiranje funkcije *f (x,y)* koja predstavlja izvornu sliku s konvolucijskom matricom ***m***. Tada je rezultantna funkcija

$$f'(x,y) = f(x,y) * \mathbf{m}[x,y]$$

Vrijednosti u konvolucijskoj matrici definiraju utjecaj susjednih piksela na određenu točku slike. Konvolucijska je matrica najčešće puno manjih dimenzija od same slike. Veće matrice možemo koristiti za otkrivanje grubljih rubova. U primjeru na slici 1, intenzitet rezultantnog piksela *y* će biti izračunata sljedećom formulom:

$$y = m_0 \cdot x_0 + m_1 \cdot x_1 + m_2 \cdot x_2 + m_3 \cdot x_3 + m_4 \cdot x_4 + m_5 \cdot x_5 + m_6 \cdot x_6 + m_7 \cdot x_7 + m_8 \cdot x_8$$



Slika 1. Operacija konvolucije

2.1.1. Otkrivanje horizontalnih rubova

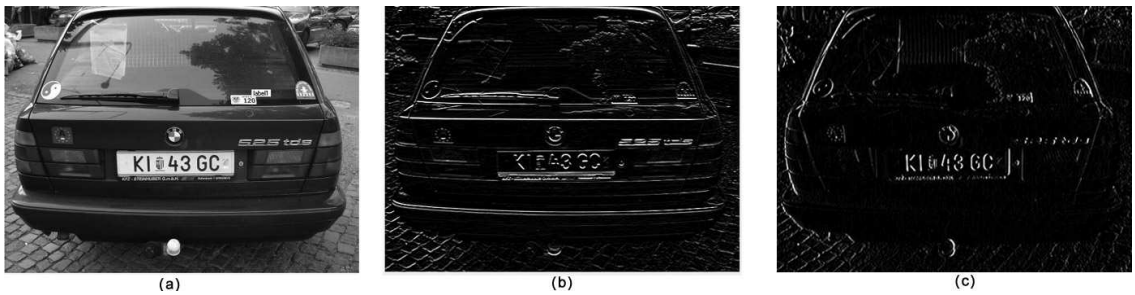
Kako bi na slici označili samo horizontalne rubove, treba konvoluirati početnu sliku sa sljedećom matricom:

$$\mathbf{m}_{hor} = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

2.1.2. Otkrivanje vertikalnih rubova

Analogno filtru za horizontalne rubove, definiramo konvolucijsku matricu za detekciju vertikalnih rubova:

$$\mathbf{m}_{ver} = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

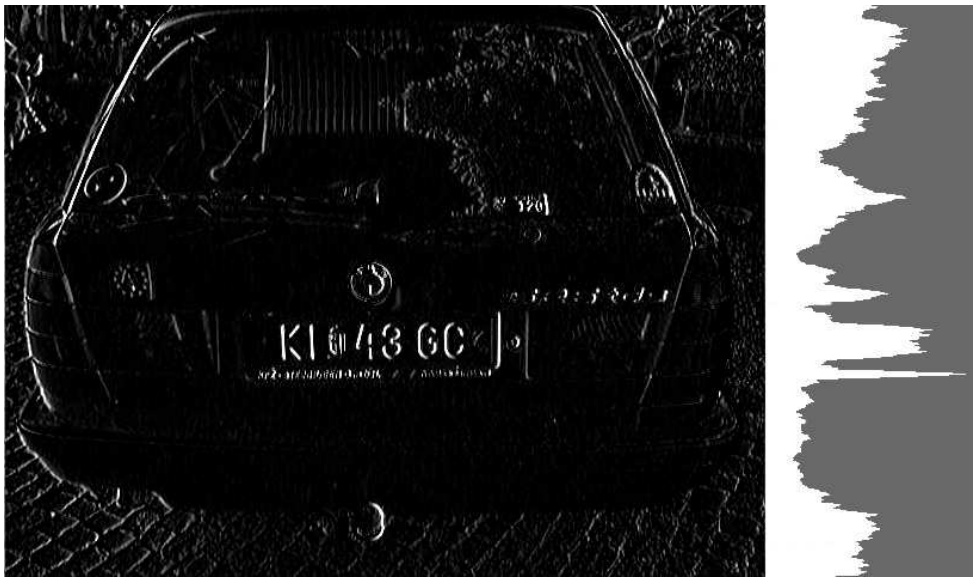


Slika 2. Rezultati obrade: (a) Originalna slika nakon pretvorbe u nijanse sive boje (b) Filtar za određivanje horizontalnih rubova (c) Filtar za određivanje vertikalnih rubova

2.2. Horizontalna i vertikalna projekcija

Nakon što smo odredili rubove na slici, preostaje nam odrediti poziciju tablice iz statističkih svojstava rezultantne slike. Jedan od načina jest projiciranje slike na koordinatne osi.

Vertikalna projekcija slike je graf koji predstavlja ukupnu težinu slike po osi *y*. Ukoliko na sliku primijenimo filtar za određivanje vertikalnih rubova, a nakon toga stvorimo vertikalnu projekciju slike, iznos projekcije u izabranoj točki će biti upravo jednak količini vertikalnih rubova u toj točki.



Slika 3. Vertikalna projekcija

Ako crno-bijelu sliku definiramo kao funkciju dvije varijable: $f(x, y)$, tada je vrijednost vertikalne projekcije u točki *y* jednaka sumi svih vrijednosti funkcije *f* u retku *y*:

$$p_y(y) = \sum_{i=0}^{w-1} f(i, y)$$

Horizontalna projekcija se definira na sličan način:

$$p_x(x) = \sum_{j=0}^{h-1} f(x, j)$$

gdje *w* i *h* predstavljaju širinu i visinu slike, respektivno.

2.3. Lokalizacija tablice

Postupak lokalizacije registarske tablice se odvija u dvije faze koje se nazivaju „izdvajanje trake“ i „izdvajanje tablice“. Prva faza podrazumijeva određivanje horizontalnog područja na slici pomoću vertikalne projekcije. U drugoj fazi se na temelju horizontalne projekcije iz trake isiječe područje koje odgovara samoj registarskoj tablici.

2.3.1. Izdvajanje trake

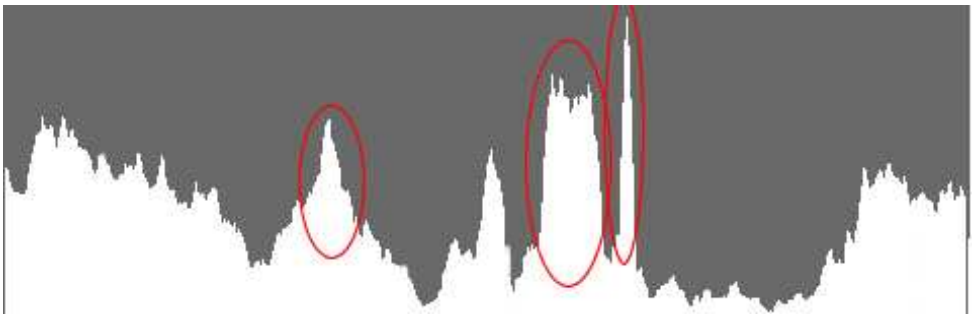
Nakon što smo odredili vertikalnu projekciju slike, potrebno je odrediti maksimume na dobivenom grafu. Svaki maksimum u vertikalnoj projekciji odgovara kandidatu za traku u kojoj se može nalaziti registarska tablica. Poziciju maksimuma izračunavamo na sljedeći način:

$$y_{t\max} = \arg\max_{y_0 \le y \le y_1} \left\{ p_y(y) \right\}$$

Maksimum odgovara centru detektirane trake. Gornja i donja granica trake se određuju formulama:

$$y_{t0} = \max_{y_0 \le y \le y_{t\max}} \left\{ y \mid p_y(y) \le p_y(y_{t\max}) \cdot c_y \right\}$$

$$y_{t1} = \min_{y_{t\max} \le y \le y_1} \left\{ y \mid p_y(y) \le p_y(y_{t\max}) \cdot c_y \right\}$$



Slika 4. Detektirani maksimumi u vertikalnoj projekciji - kandidati za traku

Konstanta c_y određuje širinu detektiranog maksimuma, a poprima vrijednosti između 0 i 1. Ukoliko je konstanta bliža jedinici, detektirana traka će biti uža. Smanjivanjem konstante povećavamo područje trake. Empirijski je utvrđeno da je optimalna vrijednost $c_y = 0.55$.

Slika 5. Izdvojena traka

Prikazani postupak se ponavlja za svaki dovoljno veliki maksimum na slici. U svakom koraku se određuju gornja i donja granica trake, te se interval između tih granica popuni nulama, kako bi se izbjegao konflikt s drugim maksimumima. Sljedeći pseudokod ilustrira ovu ideju:

```
Neka je S skup kandidata
Za svaki maksimum m iz S
     Odredi y_t0 i y_t1 analizom vertikalne projekcije
     Spremi y_t0 i y_t1 u listu L
     Popuni interval <y_t0 , y_t1> nulama
kraj
```

Konačno, lista kandidata za traku L se sortira po vrijednosti maksimuma $y_{t\max}$.

2.3.1.1. Heuristička analiza

Lista detektiranih traka će u većini slučajeva sadržavati više od jednog kandidata. Odabir prave trake možemo provesti na više načina. Prvi postupak se zasniva na slijednom odabiranju kandidata iz liste L te prosljeđivanju trake na analizu u dublje slojeve algoritma za prepoznavanje. Ukoliko proces ne uspije, vraćamo se na prvi korak i izabiremo idućeg kandidata. Mana ovog pristupa su loše performanse zbog velike vremenske složenosti algoritama za segmentaciju i klasifikaciju znakova.

Ovaj postupak možemo poboljšati korištenjem heuristike. Do sada je jedini kriterij za težinu kandidata bila maksimalna vrijednost u vertikalnoj projekciji. Ideja heuristike jest uračunati i druge faktore u ukupnu težinu svakog kandidata. Ukupnu težinu α tada možemo prikazati kao težinski zbroj svih faktora:

$$\alpha = \alpha_1 \cdot c_1 + \alpha_2 \cdot c_2 + \alpha_3 \cdot c_3$$

Svaki od faktora α_i predstavlja jednu heuristiku za evaluaciju trake, dok pripadajuće konstante određuju utjecaj pojedinog kriterija na ukupnu težinu. Kao heuristike α_i možemo uzeti sljedeće kriterije s ciljem minimizacije ukupne težine α:

Tablica 1. Heuristike za evaluaciju trake

$\alpha_1 = \lvert y_{t0} - y_{t1} \rvert$	Preferiramo trake s manjom širinom.
$\alpha_2 = \frac{1}{p_y(y_{t\max})}$	Faktor koji određuje maksimum u vertikalnoj projekciji. Preferiramo trake koje imaju veći maksimum.
$\alpha_3 = \frac{1}{\sum_{j=y_{t0}}^{y_{t1}} p_y(j)}$	Ukupna površina maksimuma ispod krivulje vertikalne projekcije, preferiramo što veću površinu.

2.3.2. Izdvajanje tablice

Opisani postupak za izrezivanje trake iz zadane slike je umnogome sličan algoritmu izdvajanja tablice iz trake. Na početku primijenimo filtar detekcije vertikalnih rubova na traci, te stvorimo horizontalnu projekciju dobivene slike. Lokalizacija tablice se tada svodi na traženje maksimuma u horizontalnoj projekciji.

$$p_x(x) = \sum_{j=y_{t0}}^{y_{t1}} f(x, j)$$

Primijetimo da ovdje radimo s horizontalnom projekcijom detektirane trake, a ne cijele slike. Ovo postižemo restrikcijom intervala sumiranja na granice $\langle y_{t0}, y_{t1} \rangle$. Kako je statistička disperzija vrijednosti u horizontalnoj projekciji često jako visoka, radimo konvoluciju s rang matricom.

Slika 6. Horizontalna projekcija trake nakon konvolucije s rang matricom. Crvenom bojom su označena mjesta gdje je detektiran rub tablice.

Maksimum u horizontalnoj projekciji određujemo na sljedeći način:

$$x_{t\max} = \arg\max_{x_0 \le x \le x_1} \{p_x(x)\}$$

Krajeve registarske tablice x_{t0} i x_{t1} određujemo slično:

$$x_{t0} = \max_{x_0 \le x \le x_{t\max}} \{x \mid p_x(x) \le p_x(x_{t\max}) \cdot c_x\}$$

$$x_{t1} = \min_{x_{t\max} \le x \le x_1} \{x \mid p_x(x) \le p_x(x_{t\max}) \cdot c_x\}$$

Vrijednost konstante c_x određena je empirijski i postavljena na 0.15. Smanjivanje vrijednosti uzrokuje detekciju preširokog područja, dok približavanjem konstante jedinici smanjujemo površinu detektirane tablice.

2.3.2.1. Heuristička analiza

Kao i u algoritmu detekcije traka, i postupak lokalizacije registarskih tablica može dati više mogućih kandidata. Odabir prave trake će nam biti olakšan primijenimo li heuristiku na dobivene rezultate. Primjer jedne takve heuristike jest omjer visine i širine detektirane trake. U većini zemalja taj omjer za jednoretčane registarske tablice iznosi 5. Tako možemo definirati našu heuristiku kao:

$$\alpha = \left| \frac{|x_{t0} - x_{t1}|}{|y_{t0} - y_{t1}|} - 5 \right|$$

Što je omjer širine i visine detektirane tablice bliži ispravnoj vrijednosti, težina kandidata za tablicu će biti manja.

Konačno, slika 7 prikazuje detektiranu registarsku tablicu na kraju prve faze algoritma prepoznavanja.

Slika 7. Detektirana registarska tablica

3. Segmentacija tablice

Nakon detekcije registarske tablice slijedi postupak segmentacije. Ovaj postupak je jedan od najkritičnijih dijelova cijelog procesa prepoznavanja, jer svi daljnji koraci ovise o izlazu algoritma segmentacije. Nepravilna segmentacija može uzrokovati da se dva odvojena znaka prepoznaju kao jedan, ili da se nepravilno presiječe jedan znak na njih više.

Postoji nekoliko implementacija ovog algoritma, uključujući algoritme s neuronskim mrežama, no jedan od najučinkovitijih se zasniva na poznatom principu – horizontalnoj projekciji binarne slike tablice. Binarna slika je ona koja sadrži piksele koji su isključivo crne ili bijele boje. Tablicu tad možemo prikazati kao funkciju dvije varijable, s kodomenom $\{0,1\}$:

$$f(x,y)=\begin{cases}0, \text{ ako je točka } (\mathrm{x,y}) \text{ tekst}\\ 1, \text{ ako je točka (x,y) pozadina}\end{cases}$$

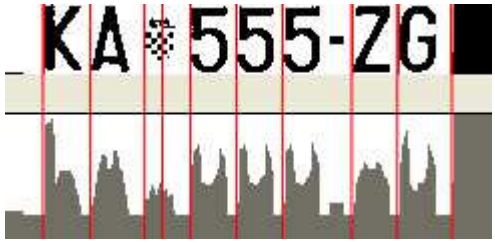
Nakon podjele tablice na horizontalne segmente slijedi korak odbacivanja elemenata koji nisu tekst. Tablica će često sadržavati nepoželjne elemente, kao što su nečistoće na samoj tablici, neravnomjerne sjene ili artefakti nastali zbog kodiranja slike. Algoritam mora odbaciti ovakve elemente i kao izlaz vratiti samo elemente koji zadovoljavaju zadane kriterije.

3.1. Binarizacija slike

Binarizacija slike (engl. *thresholding*) je postupak pretvaranja slike u sivim tonovima u sliku koja sadrži samo crne ili bijele piksele. Ovaj postupak se može provesti tako da se izabere određena vrijednost za tzv. prag (engl. *threshold*). Svi pikseli koji imaju manji intenzitet od praga se pretvaraju u crne piksele, a ostali u bijele. Postupak određivanja praga se provodi na različite načine. Najjednostavniji od njih je fiksno određivanje praga na sredini histograma. No, kod nekih slika nije dovoljno korištenje jedinstvenog praga za cijelu sliku. Nepravilne sjene na registarskoj tablici često pokvare rezultate algoritma koji koristi globalno određivanje praga, pa je potrebno koristiti adaptivne metode.

3.2. Segmentacija pomoću horizontalne projekcije

Nakon što su dobiveni zadovoljavajući rezultati pretvorbe tablice u binarni oblik (crni i bijeli pikseli) prelazimo na postupak segmentacije. Prvi korak je stvaranje horizontalne projekcije dobivene slike. Ideja algoritma počiva na principu da su znakovi na tablici linearno odvojivi. Tada između svaka dva znaka na tablici mora postojati traka pozadine (bijela boja – pikseli vrijednosti 1). S grafa vertikalne projekcije tada jednostavno očitamo maksimume i proglasimo pronađene točke granicama između znakova.



Slika 8. Horizontalna projekcija binarne slike i odgovarajuća segmentacija

Na početku moramo naći globalni maksimum na slici. On predstavlja maksimalnu količinu bijelih piksela u jednom stupcu, i koristimo ga kao parametar u traženju ostalih granica između slova. Označimo globalni maksimum s v_m. Tada je

$$v_m = \arg\max_{0 \le x \le w-1} \{p_x(x)\}$$

gdje je *w* širina tablice. Nakon toga pratimo sljedeći algoritam.

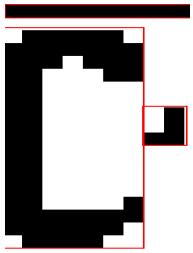
1. Pronađimo poziciju trenutnog maksimuma, nazovimo ga x_m:
$$x_m = \arg\max_{0 \le x \le w-1} \{p_x(x)\}$$
2. Odredimo lijevu i desnu granicu detektiranog razmaka:
$$x_l = \max_{0 \le x \le x_m} \{x \mid p_x(x) \le p_x(x_{t\max}) \cdot c_x\}$$
$$x_d = \min_{x_m \le x \le w-1} \{x \mid p_x(x) \le p_x(x_{t\max}) \cdot c_x\}$$
3. Popunimo interval $\langle x_l, x_d \rangle$ nulama.
4. Ukoliko je $p_x(x_m) < c_v \cdot v_m$ završi.
5. Podijeli tablicu na mjestu x_m.
6. Skoči na korak 1.

U svakoj od iteracija algoritma detektiramo po jednu granicu između znakova. Konstanta c_x određuje širinu razmaka između slova. Tipična vrijednost je 0.86. Veća vrijednost (bliža

jedinici) sužuje granicu između slova. Premala vrijednost može uzrokovati da se dio znaka tretira kao razmak. Konstanta c_v određuje granicu prihvaćanja maksimuma na projekciji kao razmaka između znakova. Tipična vrijednost ove konstante iznosi 0.9. Ukoliko postavimo veću vrijednost riskiramo da nam se neka slova ne razdvoje (poradi nepravilnosti na slici). Postavljanje niske vrijednosti može uzrokovati nepotrebno i neželjeno presijecanje znaka.

3.3. Izdvajanje znakova iz segmenata

Segmentacija tablice je tek prvi korak u pronalaženju znakova. Nakon provedene segmentacije u dosta slučajeva će se uz znak nalaziti i dijelovi koji ne pripadaju samom znaku, kao što su mrlje na registarskoj tablici, rubovi registarske tablice itd. Sljedeći problem je izdvajanje samog znaka iz segmenta registarske tablice. Za ovaj posao nam treba algoritam izdvajanja povezanih komponenti. Rezultat ovog algoritma će biti lista komponenti, a svaka od njih će se sastojati od određenog broja crnih piksela.



Slika 9. Tipičan segment tablice. Pri vrhu se vidi rub tablice, a desno jedna mrlja. Crvenim pravokutnicima su označene povezane komponente.

Cilj nam je eliminirati sve nebitne komponente tako da nam ostane samo jedna, te da ona predstavlja znak. Ukoliko se u segmentu ne nalazi niti jedno slovo, heuristika bi trebala eliminirati takvu lažnu pojavu znaka.

3.3.1. Traženje povezanih komponenti

Označimo segment funkcijom dvije varijable, $f(x, y)$. Funkcija je jednaka jedinici ukoliko je piksel na poziciji *(x,y)* crn, inače je 0.Odabiremo gornji lijevi kut kao ishodište koordinatnog sustava: *(0,0).* Donji desni kut je tada određen s *(w-1,h-1),* gdje su *w* i *h* širina i visina segmenta, respektivno. Komponenta *P* je definirana kao skup susjednih crnih piksela. Da bi ova definicija bila potpuna, moramo definirati pojam susjedstva. Kažemo da su pikseli *(x,y)* i *(x',y')* u 4-pikselnom susjedstvu ukoliko vrijedi

$$|x - x'| = 1 \oplus |y - y'| = 1$$

Ukoliko su *a* i *b* pikseli za koje vrijedi gornja formula, možemo skraćeno zapisati da vrijedi relacija $a\ddot{N}_4 b$.

Pikseli a=*(x,y)* i b=*(x',y')* su u 8-pikselnom susjedstvu ukoliko vrijedi:

$$|x-x'|=1 \vee |y-y'|=1$$

i pišemo $a\ddot{N}_8 b$. Definirajmo još i da piksel *(x,y)* pripada komponenti P ukoliko postoji piksel *(x', y')* koji pripada komponenti P i $(x,y)\ddot{N}_8(x',y')$.

Algoritam za traženje povezanih komponenti se zasniva na tzv. *poplavljivanju* (engl. *flood fill, seed fill algorithm*). Definirajmo sljedeće oznake:

- Neka je **P** povezana komponenta (skup susjednih crnih piksela)
- Neka je **S** skup svih pronađenih povezanih komponenti
- Neka je **X** skup svih crnih piksela na slici.
- Neka je **D** skup u koji upisujemo obrađene piksele, i **A** pomoćni skup

Evo i pseudokoda za dotični algoritam:

```
S=0
X=(x,y)| f(x,y)=1                          ;svi crni pikseli
Dok je skup X neprazan
     P=0;A=0;
     Uzmi jedan piksel iz X i dodaj ga u A
     Dok je skup A neprazan                ;pomoćna lista
          Uzmi jedan piksel (x,y) iz skupa A
          Ako je f(x,y)=1 ∧ D ne sadrži (x,y)
               Dodaj (x,y) u skup P
               Dodaj (x,y) u skup D        ;D – „dirty“ skup
               Ukloni (x,y) iz skupa X
               Dodaj sve piksele p za koje
          vrijedi(x,y)N̈8p u skup A
     Kraj
     Dodaj P u skup S
Kraj
```

3.3.2. Heuristička analiza povezanih komponenti

Konačno, potrebno je iz dobivenog skupa povezanih komponenti ukloniti sve segmente koji ne zadovoljavaju određene uvjete. Najjednostavniji uvjet je onaj na veličinu znaka. Možemo jednostavno izračunati prosječnu širinu i visinu detektiranih segmenata i ukloniti one komponente s dimenzijama koje značajno statistički odstupaju od prosjeka. Za heuristiku možemo uzeti i broj bijelih piksela u segmentu. Preferiramo one segmente koji imaju veći broj bijelih piksela.

Slika 10. Tablica i detektirani segmenti. Svi lažni segmenti moraju biti odbačeni.

Slika 10 ilustrira opravdanost korištenja gornjih heuristika. Rubni segmenti većinom sadrže crne piksele, te ih treba odbaciti. Odsječci koji odstupaju od prosječne širine i visine segmenta se također odbacuju.

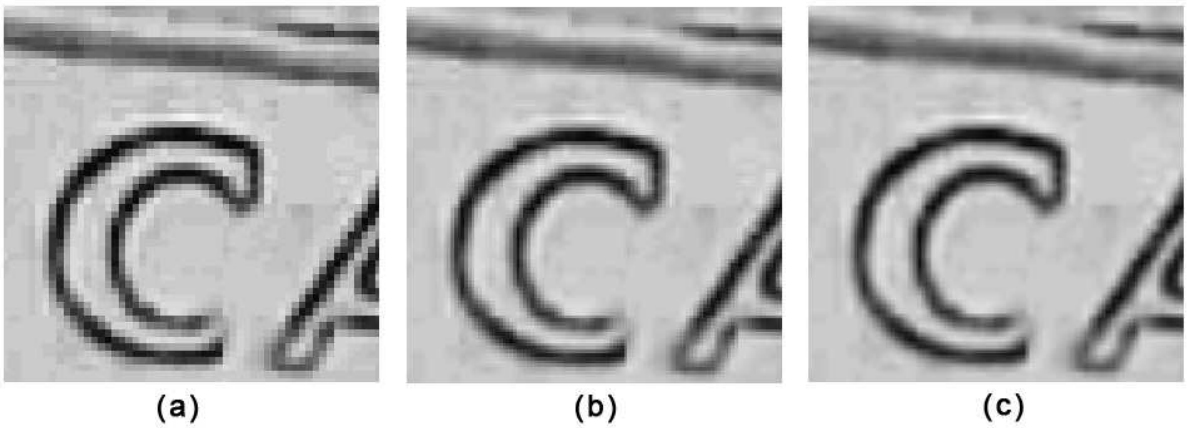
4. Analiza znakova

Kako bismo prepoznali slovo iz njegove slikovne reprezentacije moramo izdvojiti neke karakteristične značajke pogodne za postupak klasifikacije. Izabrani algoritam mora biti u mogućnosti izvući značajke koje su invarijantne na promjenu osvjetljenja, različite tipove slova i kut pogleda (rotacija ili iskošenost).
Prvi korak algoritma je normalizacija veličine slova, čime sve prepoznate znakove svodimo na istu veličinu. Nakon toga primjenjujemo algoritam izvlačenja opisnih vektora iz normaliziranih znakova.

4.1. Normalizacija veličine

Prije nego što budemo u mogućnosti izvući karakteristične značajke iz detektiranih slova, nužno ih moramo svesti na jedinstvenu veličinu. Ovaj postupak se naziva ponovno uzorkovanje (engl. *resampling*). U većini slučajeva ćemo htjeti smanjiti veličinu slova. Taj postupak podrazumijeva odbacivanje dijela informacije iz originalne slike.

Postoji više algoritama za promjenu veličine slike. Najpoznatiji su algoritam najbližeg susjeda (engl. *nearest neighbor*), bilinearno i bikubično filtriranje. U praksi se pokazuje da za posao smanjivanja dimenzije slike najbolje rezultate u prihvatljivom vremenu daje bikubično filtriranje.



Slika 11. Promjena veličine slike korištenjem različitih algoritama:
(a) najbliži susjed, (b) bilinearno filtriranje, (c) bikubično filtriranje

4.2. Izvlačenje značajki

Slikovna reprezentacija znaka nije pogodna za obradu i klasifikaciju. Čak i mala promjena pozicije znaka može uzrokovati veliku razliku u količini i poziciji crnih piksela u slici. Zato je potrebno pronaći način za opisivanje znaka pomoću karakteristika koje se mogu lako detektirati, a da jedinstveno opisuju pojedini znak. Idealno bi bilo kad bi te karakteristike bile invarijantne na poziciju znaka, njegovu rotaciju, te tip slova koji se koristi na registarskim tablicama. Konačni rezultat algoritma izvlačenja značajki (engl. *feature extraction algorithm*) će za svaki znak generirati tzv. vektor opisnika (deskriptora):

$$\mathbf{x} = \left[x_0, x_1, \ldots x_{n-1} \right]$$

Svaki od elemenata ovog vektora predstavlja jednu karakterističnu značajku detektiranog znaka. U većini slučajeva se pokazuje da se znak može najpreciznije opisati svojim oblikom, tj. brojem linija, petlji, krajeva linija, krivulja itd. Postupak određivanja ovih značajki zove se strukturalna analiza.

4.2.1. Algoritam skeletonizacije

Sirovi oblik slova nakon određivanja povezanih komponenti nije prikladan za detekciju karakterističnih značajki. Potrebno je svesti znak na njegov "kostur", tj. prikazati ga u obliku grafa. Dakle, moramo odbaciti sve crne piksele koji ne nose dodatnu informaciju o obliku slova. Klasičan postupak erozije nije dovoljan jer bi se krajevi linija mogli izbrisati, čime se gubi informacija. Za ovu svrhu razvijeni su posebni algoritmi skeletonizacije koji se zasnivaju na matematičkom principu određivanja medijalne osi.

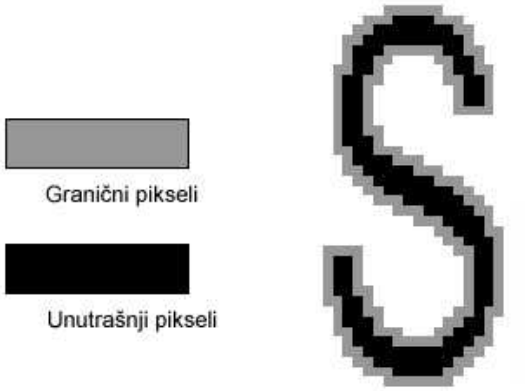
Na početku moramo definirati neke osnovne pojmove. U poglavlju 3.3 je dan pojam susjedstva:

$$\text{4-pikselno susjedstvo: } \left|x - x'\right| = 1 \oplus \left|y - y'\right| = 1$$

$$\text{8-pikselno susjedstvo: } \left|x - x'\right| = 1 \vee \left|y - y'\right| = 1$$

Za potrebe algoritma skeletonizacije definiramo skup piksela G, koji predstavlja granicu znaka. Piksel je granični, dakle pripada u skup G ako je crn i ako ima barem jednog susjeda u 8-pikselnom susjedstvu:

$$(x, y) \in G \Leftrightarrow (f(x, y) = 0 \wedge \exists (x', y') : f(x, y) = 1 \wedge (x, y) \ddot{N}_8 (x', y')$$



Slika 12. Granični i unutarnji pikseli slova

Skup unutrašnjih piksela **U** definiramo kao sve piksele koji su crni a ne pripadaju skupu **G**:

$$(x, y) \in U \Leftrightarrow (f(x, y) = 0 \wedge (x, y) \notin G)$$

Znak **Z** tad definiramo kao skup graničnih i unutrašnjih piksela, $\mathbf{Z} = \mathbf{G} \in \mathbf{U}$.

Medijalna transformacija se definira kako slijedi. Za svaki unutrašnji piksel p_u treba pronaći najbliži granični piksel p_g. Ako piksel p_u ima više takvih susjeda, kažemo da pripada središnjoj (medijalnoj) osi znaka, tj. njegovom kosturu. Ako skelet označimo kao skup piksela S, možemo zapisati sljedeću formalnu definiciju:

$$p \in S \Leftrightarrow \exists p_1 \exists p_2 : p_1 \in B \wedge p_2 \in B \wedge d(p, p_1) = d(p, p_2) = \min_{p' \in B} \{d(p, p')\}$$

Dakle, piksel *p* pripada medijalnoj osi znaka ako postoje dva piksela koji pripadaju graničnom skupu, a udaljenost do njih je minimalna udaljenost do graničnog skupa **G**.

Postupak skeletonizacije se može jednostavno predočiti konceptom vatrene fronte. Zamislimo da je zapaljena vatra na granici znaka i okoline, i da sve vatrene fronte napreduju istom brzinom. Tada je skelet znaka onaj skup piksela do kojih je vatra došla s više strana istovremeno.

Algoritam se zasniva na višestrukim prolazima kroz petlju. Petlja se sastoji od dva dijela. U prvom dijelu se na temelju određenih kriterija označavaju pikseli koji se trebaju izbrisati. Drugi korak briše izabrane piksele. Postupak se ponavlja dok od znaka nije ostao samo skelet, tj. dok prvi korak ne označi niti jedan piksel.

Uvjeti koji određuju da li će piksel biti označen u prvom koraku su sljedeći:

1) Gornji, desni ili donji susjed piksela mora biti bijeli

$$p_{gornji} \notin \mathbf{Z} \vee p_{desni} \notin \mathbf{Z} \vee p_{donji} \notin \mathbf{Z}$$

2) Lijevi, desni ili donji susjed piksela mora biti bijeli

$$p_{lijevi} \notin \mathbf{Z} \vee p_{desni} \notin \mathbf{Z} \vee p_{donji} \notin \mathbf{Z}$$

3) Piksel mora imati barem 2, a najviše 6 crnih susjeda. Ovaj uvjet sprječava brisanje krajeva linija i prekidanje spojnosti znaka

4) Ako idemo u krug po 8-pikselnom susjedstvu:

$$p_{gornji}, p_{gornji-desni}, p_{desni}, p_{donji-desni}, p_{donji}, p_{donji-lijevi}, p_{lijevi}, p_{gornji-lijevi}, p_{gornji}$$

broj prijelaza bijelo-crno mora biti jednak 1.

Nakon što su svi pikseli koji zadovoljavaju uvjete označeni u prvom koraku, drugi korak ih definitivno briše. Svaki korak iterativno stanjuje znak.

Sljedeći pseudokod formalno prikazuje opisani algoritam:

```
Čini
     nastaviti=0
     Za svaki piksel p iz skupa Z           ;cijeli znak
          Ako p ima jednog bijelog susjeda
               Dodaj p u G                  ;granični pikseli
     Kraj
     Za svaki piksel p iz skupa G
          Ako p zadovoljava uvjete 1-4
               Označi p za brisanje
               nastaviti=1
     Kraj
     Za svaki piksel p iz skupa G
          Ako je p označen za brisanje
               Izbriši p
     Kraj
Dok nastaviti=1
```

Slika 13. Segmentirana ploča i prepoznati znakovi nakon algoritma skeletonizacije

4.2.2. Strukturalna analiza

Strukturalna analiza je metoda izvlačenja karakteristika iz znakova koja polazi od pretpostavke da se slovo može predstaviti strukturalnim značajkama, poput krajeva linija, petlji i sjecišta.

4.2.2.1. Karakteristike

Kraj linije

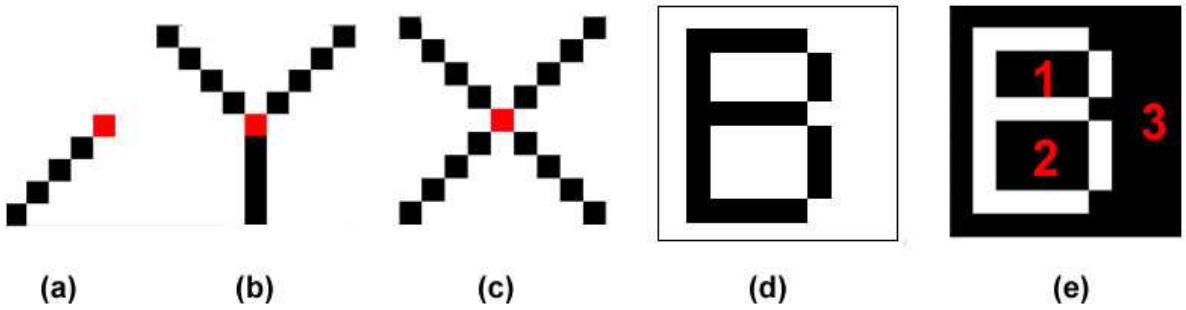
Kraj linije je piksel koji ima samo jednog crnog susjeda.

Sjecište

Sjecište je piksel koji ima barem 3 susjeda u 8-pikselnom susjedstvu. Postoje dvije klase sjecišta: sjecište triju ili četiriju linija.

Petlja

Broj petlji u skeletoniziranom znaku se određuje na malo teži način. Prvo se napravi inverzna slika znaka. Tada se prebroje sve crne povezane komponente. Od dobivenog broja moramo oduzeti 1 (jer smo brojili i crnu pozadinu) da bi dobili broj petlji u znaku.



Slika 14. (a) Kraj linije (b) Sjecište 3 linije (c) Sjecište 4 linije
(d) Slovo s 2 petlje (e) Označene povezane komponente nakon inverzije

4.2.2.2. Klase znakova

S obzirom na navedene karakteristike, sve znakove na registarskim tablicama možemo svrstati u klase. Neki znakovi imaju više obličja, pa spadaju u više klasa. Tablica prikazuje podjelu znakova po klasama.

Tablica 2. Klase znakova po karakteristikama

	Krajevi linija	**Sjecišta**	**Petlje**
0	B,8,D,O,0	C,G,I,J,L,S,U,V,Z,1,5,7,D,O,0	C,G,I,J,L,S,U,V,Z,1,5,7,E,F,T,Y,3,4,2,H,K,X,N,M,W,M
1	P,Q,6,9	E,F,T,Y,3,4,2,P,Q,6,9	A,4,R,D,O,0,P,Q,6,9
2	R,4,C,G,I,J,L,S,U,V,Z,1,5,7	R,4,B,8, H,K,X,N,M	B,8
3	A,4 ,E,F,T,Y,3,4	A,4,W,M	
4	H,K,X,N,M		
5	W,M		

Ukoliko stvorimo vektore opisnika pomoću ove tri značajke:

x=[krajevi linija, sjecišta, petlje]

dobivamo devet klasa znakova (klase nisu disjunktne radi polimorfizma slova):

Tablica 3. Vektori deskriptora i pripadajući znakovi

x	**Pripadajući znakovi**
[3,3,1]	A, 4
[2,2,1]	R, 4
[0,2,2]	B, 8
[2,0,0]	C, G, I, J, L, S, U, V, Z, 1, 5, 7
[0,0,1]	D,O,0
[3,1,0]	E, F, T, Y, 3, 4, 2
[4,2,0]	H, K, X, N, M
[1,1,1]	P, Q, 6, 9
[5,3,0]	W, M

U ovom trenutku imamo izbor: prvi pravac djelovanja je da ne radimo podjelu znakova po grupama, već da vektore opisnika šaljemo u klasifikator, a zatim dobivene rezultate prihvaćamo ili odbacujemo u ovisnosti o tome da li prepoznati znak zadovoljava gore prikazana strukturalna ograničenja. Drugi pristup problemu je da svaki znak već nakon strukturalne analize pridijelimo odgovarajućoj grupi, te da za svaku grupu napravimo poseban klasifikator. Drugi način je kompleksniji za izvedbu, no u većini slučajeva daje bolje rezultate.

U našem konkretnom slučaju primjećujemo da je grupa znakova opisana s vektorom [2,0,0] najbrojnija, te da njezin vektor nosi najmanje informacija. Za razliku od ostalih grupa nad kojima možemo provesti jednostavan postupak pozicijske klasifikacije (opisan u narednom poglavlju), nad ovom grupom ćemo primijeniti klasifikaciju putem neuronske mreže. U tu svrhu moramo redefinirati naš vektor opisnika tako da nosi što je više moguće korisnih informacija pogodnih za postupak klasifikacije.

Prvi korak je da u vektor opisnika zapišemo koordinate krajeva linija. Različita slova imaju završetke na različitim dijelovima znaka, što će omogućiti neuronskoj mreži da ih klasificira. Prvo je potrebno normalizirati (*x*,*y*) koordinate točke na raspon $\langle -1,1 \rangle$. Nakon toga određujemo polarne koordinate (r,φ) karakterističnih točaka:

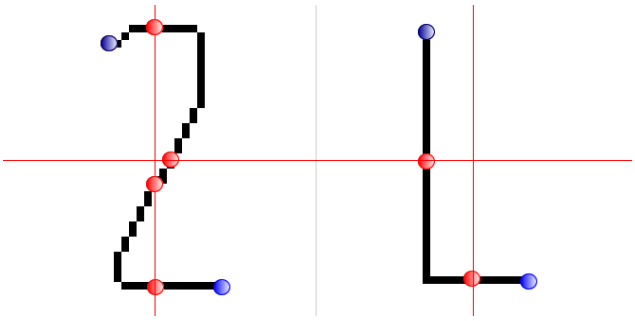
$$r = \sqrt{x'^2 + y'^2} \quad \varphi = arctg(\frac{y'}{x'}) \quad x' = \frac{2 \cdot x - w}{w} \quad y' = \frac{2 \cdot y - h}{h}$$

Ovdje su x' i y' normalizirane koordinate, a *w* i *h* širina i visina znaka, respektivno.

Prikazanim postupkom dobivamo vektor opisnika koji se sastoji od 4 člana:

$$\mathbf{x} = [r_1, \varphi_1, r_2, \varphi_2]$$

Taj vektor možemo direktno proslijediti neuronskoj mreži, ili ga pak proširiti dodatnim informacijama. Neki od najčešćih postupaka određivanja dodatnih značajki su brojanje sjecišta s određenim brojem vertikalnih i horizontalnih linija:



Slika 15. Slova Z i L imaju krajeve linija na sličnim mjestima (plavi kružići). Broj sjecišta s horizontalnim i vertikalnim linijama povučenim po sredini znaka se značajno razlikuje (crveni kružići).

U vektor tada dodajemo broj sjecišta znaka i zamišljenih linija. Dobivamo vektor sa šest elemenata:

$$\mathbf{x} = [r_1, \varphi_1, r_2, \varphi_2, n_{hor}, n_{ver}]$$

5. Klasifikacija znakova

U prošlom poglavlju bilo je riječi o izvlačenju karakterističnih značajki iz detektiranih znakova. U ovom poglavlju će biti opisano kako klasifikator na temelju ulaznih opisnih vektora raspoznaje različite znakove.

5.1. Opći problem klasifikacije

Opći problem klasifikacije se bavi preslikavanjem elemenata između dva skupa. Neka skup A sadrži sve moguće kombinacije deskriptora, a skup B sve klase znakova. Tada definiramo klasifikaciju kao preslikavanje grupe sličnih elemenata iz skupa A u zajedničku klasu koja je predstavljena jednim elementom skupa B. Dakle, jedan element u skupu B odgovara jednoj klasi.

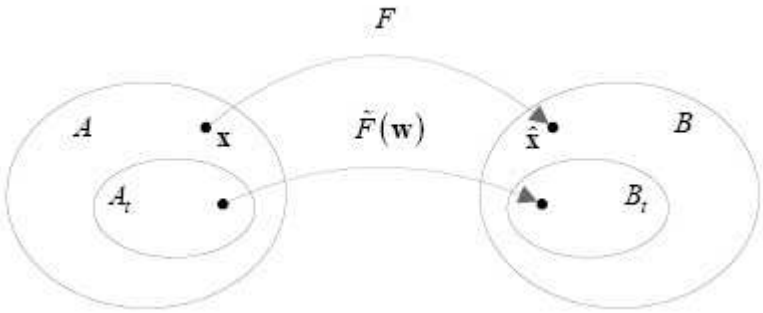
Neka je *F* hipotetska funkcija koja preslikava svaki element iz skupa A u skup B.

$$F : A \rightarrow B$$
$$\hat{\mathbf{x}} = F(\mathbf{x})$$

gdje je **x** vektor deskriptora (ili uzorak) koji reprezentira strukturu znaka, a $\hat{\mathbf{x}}$ klasifikator, koji predstavlja semantiku znaka. Funkcija *F* je najbolji teoretski klasifikator, jer "poznaje" sve elemente iz skupa A i zna ih preslikati u skup B. No, u praksi je veoma neučinkovito, ili čak nemoguće, odrediti sve moguće kombinacije deskriptora. Zato konstruiramo klasifikator iz podskupa svih preslikavanja iz A u B. Ovaj skup se naziva trenirajućim skupom. Iz ograničenog skupa preslikavanja pokušavamo konstruirati aproksimaciju funkcije *F:*

$$\tilde{F}(\mathbf{w}) : A_t \rightarrow B_t$$
$$\hat{\mathbf{x}} = \tilde{F}(\mathbf{x}, \mathbf{w})$$

gdje je **w** parametar koji utječe na kvalitetu aproksimacije, $\mathbf{x} \in A_t \subset A$, $\hat{\mathbf{x}} \in B_t \subset B$. Formalno možemo reći da je $\tilde{F}$ restrikcija funkcije *F* na skup A_t. Pri tome pretpostavljamo da za svaki $\mathbf{x}_i \in A_t$ znamo željenu vrijednost $\hat{\mathbf{x}}_i \in B_t$.



Slika 16. Preslikavanje između skupova A i B. F je hipotetska funkcija koja preslikava svaku moguću kombinaciju ulaznog uzorka $\mathbf{x}$ u odgovarajuću klasu $\hat{\mathbf{x}}$. Ova funkcija je aproksimirana funkcijom $\tilde{F}(\mathbf{w})$ koja preslikava ulaze iz skupa za treniranje A_t u odgovarajuću klasu skupa B_t.

Problem leži u pronalaženju optimalne vrijednosti parametra **w**. Taj parametar je uglavnom vektor sintaktičkih težina u neuronskoj mreži. Mijenjajući parametar w pokušavamo približiti vrijednosti funkcije $\tilde{F}(\mathbf{x}, \mathbf{w})$ što je moguće bliže vrijednostima funkcije *F*. Kako bi evaluirali vrijednost parametra **w** definiramo funkciju greške:

$$E(\mathbf{w}) = \frac{1}{2}\sum_{i=0}^{m-1}(\tilde{F}(\mathbf{x}_i, \mathbf{w}) - \hat{\mathbf{x}}_i)^2$$

gdje je *m* broj uzoraka u skupu za treniranje A_t. Neka je $\mathbf{w}_+$ optimalna vrijednost parametra **w**, dakle $\mathbf{w}_+ = \arg\min_{w \in W}\{E(\mathbf{w})\}$. Tada aproksimaciju $\tilde{F}(\mathbf{x}, \mathbf{w}_+)$ smatramo prilagođenom. Aproksimacija simulira originalnu funkciju za uzorke iz skupa za treniranje. Uz to, ova aproksimacija može predvidjeti izlazni klasifikator $\hat{\mathbf{x}}$ za nepoznati uzorak $\mathbf{x}$ iz skupa za testiranje A_x ($A_x = A - A_t$). Funkcija s takvom sposobnošću predviđanja djelomično zamjenjuje hipotetski klasifikator F(**x**).

5.2. Pozicijska analiza

Ako se odlučimo za princip grupiranja znakova po grupama u fazi strukturne analize, primijetit ćemo da neke grupe imaju mali broj članova (neke samo dva znaka!). Iako je moguće za svaku od njih napraviti posebnu neuronsku mrežu, puno je brže i učinkovitije iskoristiti neki od jednostavnijih algoritama kako bi razlikovali znakove unutar pojedine grupe. Pozicijska analiza spada u takve algoritme.

Uzmimo za primjer grupu definiranu opisnikom [1,1,1], dakle znakovi koji sadrže točno jedan kraj linije, jedno sjecište i jednu petlju. U tu grupu spadaju znakovi „P“, „Q“, „6“ i „9“. Pozicijska analiza od nas traži da znamo točne koordinate svake karakteristike. Krajevi linija i sjecišta su određena svojom pozicijom, dok je petlja određena pozicijom svojeg središta. Sada možemo napisati jednostavan klasifikator koji će razlikovati ova četiri slova.

```
Ako je kraj linije iznad središta
     Znak=“6“
     Inače
          Ako je kraj linije desno od središta
               Znak=“Q“
          Inače
               Ako je sjecište desno od središta
                    Znak=“P“
               Inače
                    Znak=“9“
```

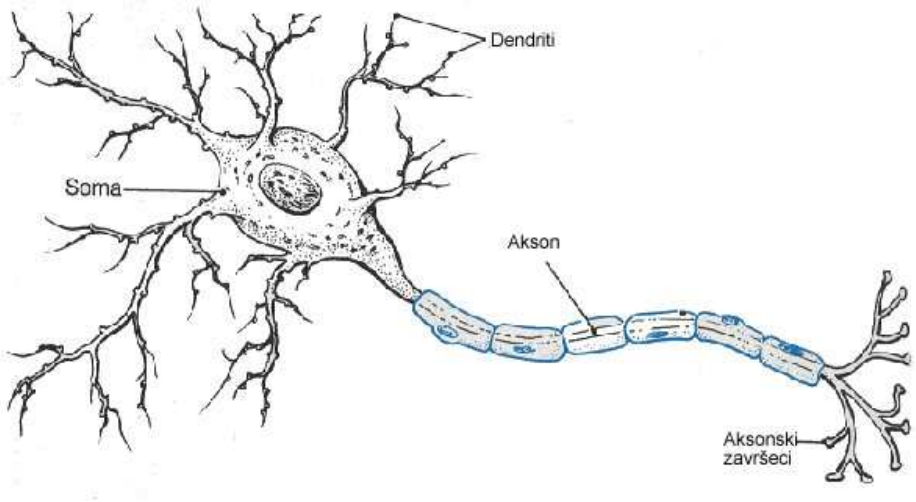
Slične klasifikatore možemo napisati za svaku od grupa koji imaju relativno malen broj znakova i dovoljan broj karakteristika. No za klasu [2,0,0] ovaj pristup nije primjeren i najbolji izbor predstavlja neuronska mreža.

5.3. Neuronske mreže

U ovom radu klasifikator je realiziran jednostavnim modelom višeslojne neuronske mreže bez povratnih veza uz povratnu propagaciju pogreške (engl. *feed-forward* , *backpropagation learning*). U nastavku su izloženi osnovni principi neuronskih mreža te njihova primjena na proces klasifikacije znakova.

5.3.1. Biološki neuron

Mozak je skup od otprilike 10 milijardi međusobno povezanih neurona. Neuron je stanica koja koristi biokemijske reakcije kako bi primala, obrađivala i prosljeđivala informacije. Svaki neuron sadrži tisuće ulaznih spojeva zvanih *dendriti*. Također sadrži i jedan dugi izlazni spoj zvan *akson*, koji može biti dugačak i do nekoliko metara. Signali se kroz mozak prenose kao električni signali uzduž aksona. Kada signal stigne do *sinapse,* tj. spoja aksona i druge stanice, uzrokuje ispuštanje kemijskih spojeva (neurotransmitera) u odgovarajući dendrit.



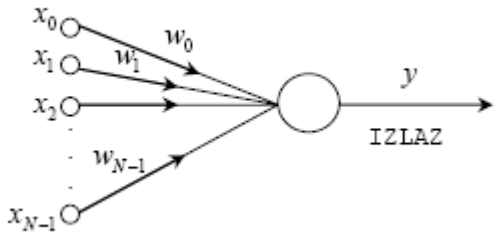
Slika 17. Biološki neuron

Postoje dvije vrste sinaptičkih veza: pobuđujuće i inhibicijske. Prva vrsta pojačava primljeni signal, dok ga druga guši. Ponašanje veze je određeno njenom „težinom“. Neuronska mreža sadrži mehanizme pomoću kojih može promijeniti težine svojih spojeva. Dakle, ljudsko pamćenje se može predstaviti sustavom težina sinaptičkih veza. Ljudi uče mijenjajući postojeće veze, čime se stare informacije malo-pomalo gube.

5.3.2. Matematički modeli neurona

5.3.2.1. McCulloch-Pitts model neurona

Jedan od prvih predloženih modela umjetnog neurona bio je McCulloch-Pitts model neurona s binarnim pragom. Neuron ima samo dva moguća izlaza (0 ili 1), i dvije vrste sinaptičkih ulaza: potpuno pobuđujuće i potpuno inhibicijske. Pobuđujuća težina ne utječe na ulaz (množi se s 1), ali ga inhibicijska negira (množenje s -1).



Slika 18. Jednostavni model neurona

Izlaz neurona je opisan funkcijom $y = g(\sum_{i=0}^{N-1} w_i \cdot x_i - \vartheta)$, gdje su *w* ulazne težine, *x* ulazi, a ϑ prag neurona. Funkcija *g* se naziva aktivacijskom funkcijom. U ovom slučaju radi se o jednostavnoj funkciji praga:

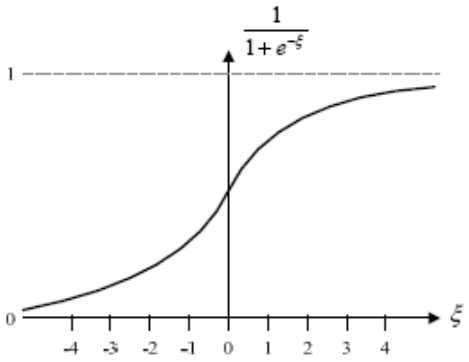
$$g(x) = \begin{cases} 0, x < 0 \\ 1, x \geq 0 \end{cases}$$

McCulloch i Pitts su pokazali da sinkrono polje ovakvih neurona može realizirati bilo kakvu računalnu funkciju, slično Turingovom stroju. No, kako biološki neuroni nemaju binarni odziv, već kontinuirani, ovaj model neurona nije prikladan za aproksimaciju bioloških neuronskih mreža.

5.3.2.2. Perceptron

Perceptron se nadovezuje na ideju binarnog neurona, no ulazne težine više nisu ograničene na 0 i 1, već mogu poprimiti proizvoljne vrijednosti. Analogno tome, i izlaz neurona je kontinuirana funkcija. Daljnje poboljšanje modela je realizirano zamjenom aktivacijske funkcije praga *g* sa sigmoidnom aktivacijskom funkcijom.

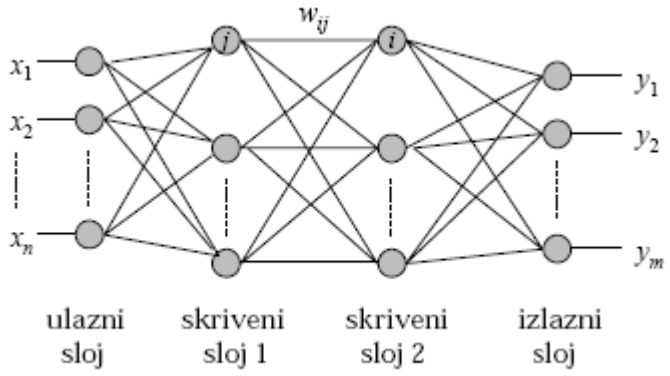
$$y = g(\sum_{i=0}^{N-1} w_i \cdot x_i - \vartheta), \qquad g(x) = \frac{1}{1+e^{-x}}$$



Slika 19. Sigmoidna aktivacijska funkcija

5.3.3. Višeslojna neuronska mreža s unaprijednom propagacijom

Višeslojne neuronske mreže s unaprijednom propagacijom (engl. *multilayer feedforward networks*) predstavljaju važnu klasu neuronskih mreža. Tipično, mreža se sastoji od skupa senzornih elemenata (ulazni čvorovi) koji čine ulazni sloj, jednog ili više skrivenih slojeva procesnih elemenata, i izlaznog sloja procesnih elemenata. Ulazni signal širi se mrežom unaprijed, sloj po sloj. Ove se neuronske mreže obično nazivaju višeslojni perceptroni (engl. *multilayer perceptrons – MLPs*).



Slika 20. Višeslojni perceptron s jednim ulaznim, dva skrivena i jednim izlaznim slojem

Višeslojni perceptroni se uspješno primjenjuju za rješavanje raznolikih problema i to kroz postupke nadgledanog učenja (učenje s učiteljem) algoritmom povratne propagacije pogreške (engl. *error back-propagation algorithm*). Algoritam se temelji na pravilu učenja korekcijom greške. U osnovi, učenje povratnom propagacijom greške sastoji se od dva prolaza kroz različite slojeve mreže: prolazak unaprijed i prolazak unatrag. U prolasku unaprijed, aktivni uzorak (ulazni vektor) postavlja se na ulazne čvorove mreže, i njegov učinak se širi kroz svaki sloj mreže. Konačno, generira se skup izlaza kao konkretan odziv mreže. Tijekom prolaska unaprijed sve sinaptičke težine mreže su fiksne, nepromjenjive. Tijekom prolaska unatrag, sinaptičke težine se podešavaju u skladu s pravilom učenja korekcijom greške. Detaljnije, razlika željenog (ciljanog) odziva i trenutnog konkretnog odziva mreže predstavlja signal greške. Taj se signal greške širi unatrag kroz mrežu, u suprotnom smjeru od sinaptičkih veza – otuda i ime "povratna propagacija greške". Sinaptičke težine podešavaju se tako da realni odziv mreže bude bliže željenom odzivu u statističkom smislu. Proces učenja koji se izvodi u algoritmu naziva se učenje povratnom propagacijom (engl. *back-propagation learning*).

5.3.4. Primjena neuronske mreže na klasifikaciju znakova

Proces korištenja neuronske mreže kao klasifikatora znakova se odvija u dvije faze. Prva faza podrazumijeva stvaranje neuronske mreže i njezino treniranje. U drugoj fazi se istrenirana mreža koristi kako bi klasificirala prepoznate znakove.

5.3.4.1. Faza treniranja

Na početku je potrebno stvoriti samu neuronsku mrežu. Za primjenu u algoritmima klasifikacije znakova koristit ćemo troslojni model perceptrona s unaprijednom propagacijom. Broj neurona u svakom sloju se određuje iz konteksta u kojem želimo koristiti neuronsku mrežu.

Ulazni sloj ima onoliko neurona kolika je duljina karakterističnog vektora koji opisuje pojedini znak. Na kraju četvrtog poglavlja smo ustanovili dužinu karakterističnog vektora, koja iznosi 6. Dakle, ulazni sloj neuronske mreže će imati šest neurona.

Broj neurona u skrivenom sloju je proizvoljan. Metode otkrivanja optimalne vrijednosti su veoma kompleksne i zahtjevne za implementaciju. Premali broj neurona u skrivenom sloju može uzrokovati da neuronska mreža ne bude u mogućnosti učiti nove uzorke. Preveliki broj neurona uzrokuje prekomjernu naučenost mreže, koja ne može ispravno generalizirati nepoznate uzorke. Empirijski podaci su pokazali da se

zadovoljavajući rezultati dobivaju ako je broj neurona u skrivenom sloju jednak broju ulaznih ili izlaznih neurona, ili njihova aritmetička sredina.

Izlazni sloj se sastoji od onoliko neurona koliko postoji klasa znakova. Ukoliko želimo razlikovati *n* znakova, toliko ćemo neurona stvoriti u izlaznom sloju. Konkretno, grupa znakova opisana vektorom [2,0,0] iz poglavlja 4 sadrži 12 znakova, dakle stvaramo dvanaest izlaznih neurona.

Treniranje neuronske mreže započinje definiranjem skupa podataka za učenje. Za svaki znak koji želimo prepoznati definiramo ulazni vektor, kao i željeni izlaz. Ulazni vektor određujemo tako da provedemo postupak strukturne analize nad idealnim primjerkom znaka. Izlazni vektor opisuje željeni odziv neuronske mreže. Ako smo definirali aktivacijsku funkciju kao običnu bipolarnu sigmoidu, tada će se odziv neuronske mreže nalaziti u intervalu $\langle -0.5, 0.5 \rangle$. Izlazni vektor za znak *i* stvaramo tako da na poziciju *i* izlaznog vektora upišemo vrijednost 0.5, a na sva ostala mjesta vrijednost -0.5.

Nakon što smo definirali ulazne i izlazne vektore za svaki znak, započinjemo postupak nadgledanog učenja mreže. Učenje je podijeljeno u više epoha. Unutar svake od njih algoritam za učenje (engl. *teacher*) prođe kroz sve definirane ulaze i izlaze i prilagodi težine veza između neurona metodom povratne propagacije greške. Nakon svake epohe učitelj kao povratnu vrijednost vrati iznos pogreške. Pogreška se određuje na temelju razlike željenog izlaza neuronske mreže i trenutnog odziva mreže.

Općenito vrijedi pravilo da mreža povećanjem broja epoha smanjuje izlaznu pogrešku. Međutim, ne može se pokazati da će greška uvijek konvergirati u nulu. Zato se u većini primjena promatra gradijent greške te se učenje prekida nakon što gradijent padne ispod neke zadane vrijednosti.

Dva su parametra koja možemo mijenjati kod nadgledane metode učenja: *brzina učenja* i *moment*. Brzina učenja određuje veličinu promjene težine veza između neurona pri svakom prolazu algoritma. Male vrijednosti ovog parametra uzrokuju male promjene težina, i obrnuto. Odabir najbolje vrijednosti za brzinu učenja nije očit. Ako postavimo brzinu učenja na 0, mreža neće uopće učiti. Moment je faktor koji određuje koliko će utjecaja prethodne promjene težina imati na trenutnu promjenu. Najčešće se koristi kako bi "izgladili" gradijent učenja. (Refenis, 1994) je u svom radu iznio podatak da neuronske mreže s iznosom brzine učenja od 0.2 i momenta između 0.4 i 0.5 daju najbolje

performanse pri konvergenciji. Primijećeno je (Jacobs, 1988) kako se puno bolji rezultati dobivaju ako se brzini učenja dozvoli da varira u vremenu. Brzinu učenja povećavamo ako je derivacija promatranog parametra istog predznaka tijekom duljeg vremenskog razdoblja. Ukoliko derivacija mijenja predznak, trebamo “izgladiti” skokove smanjenjem brzine učenja.

5.3.4.2. Faza klasifikacije

U ovom trenutku imamo istreniranu neuronsku mrežu spremnu za raspoznavanje znakova. Sve što je potrebno napraviti jest proslijediti opisni vektor na ulaz neuronske mreže i iščitati izlazni vektor. U idealnom slučaju, jedan element vektora bi imao vrijednost 0.5, a svi ostali -0.5. Kako se to u realnim uvjetima ne može postići, trebat ćemo proći kroz čitav vektor i naći maksimalnu vrijednost. Taj element uzimamo kao prepoznati znak. Dobra praksa je zapamtiti nekoliko elemenata sa najvišim vrijednostima i sortirati ih po težini. Ukoliko se dogodi da sintaksna analiza odbaci određeni element možemo jednostavno uzeti sljedeći element iz polja i predati ga na ponovnu analizu.

6. Sintaksna analiza

U nekim situacijama kada proces prepoznavanja ne uspije ispravno prepoznati znak, postoji mogućnost otkrivanja i ispravljanja greške pomoću sintaksne analize. Ako imamo skup pravila za pojedinu državu, možemo ocijeniti ispravnost prepoznatih znakova.

Ako se, na primjer, algoritam za prepoznavanje ne može odlučiti između znakova „0“ i „O“, konačna odluka će biti donesena na temelju sintaksnog uzorka. Ukoliko je u uzorku na tom mjestu predviđena brojka, bit će izabran znak „0“. Ako je pak predviđeno slovo, izabrat će se znak „O“.

6.1. Sintaksni uzorci

U praksi, sustav za raspoznavanje registarskih tablica mora biti u mogućnosti prepoznavati tablice iz različitih zemalja. Kako ne postoji jedinstveno pravilo po kojem bi se formirale sve tablice, sustav treba poznavati više sintaksnih uzoraka. Jedan od ključnih problema jest i odlučivanje koji uzorak treba primijeniti u danoj situaciji. Dakle, sustav mora prvo prepoznati tip registarske tablice, a zatim primijeniti odgovarajući sintaksni uzorak. Neki komercijalni sustavi čak pokušavaju pronaći naljepnicu s oznakom države kako bi riješili ovaj problem.

Formalno, ako je registarska tablica skup od *n* znakova $T = (p_0, p_1, ... p_{n-1})$, tada je sintaksni uzorak *n*-torka koja sadrži skupove znakova koji se mogu pojaviti na *i*-tom mjestu : $\mathbf{T}' = (\mathbf{p}'_0, \mathbf{p}'_1, ... \mathbf{p}'_{n-1})$.

Na primjer, hrvatske registarske tablice se sastoje od dvije slovne oznake koje određuju registracijsko područje (npr. ZG, ZD, ST...), zatim tri ili četiri znamenke, te na kraju jedna ili dvije slovne oznake koje ne sadržavaju dijakritičke znakove. Dakle, možemo definirati tri sintaksna uzorka:

$$\mathbf{T}'_1 = \left\{\{A-\check{Z}\},\{A-\check{Z}\},\{0-9\},\{0-9\},\{0-9\},\{A-Z\}\right\}$$

Primjer: ZD 328 S

$$\mathbf{T}'_2 = \left\{\{A-\check{Z}\},\{A-\check{Z}\},\{0-9\},\{0-9\},\{0-9\},\{0-9, A-Z\},\{A-Z\}\right\}$$

Primjer: ZG 8654 D, ST 123 AB

$$\mathbf{T}'_3 = \left\{\{A-\check{Z}\},\{A-\check{Z}\},\{0-9\},\{0-9\},\{0-9\},\{0-9\},\{A-Z\},\{A-Z\}\right\}$$

Primjer: RI 4859 AH

Možemo još postaviti i dodatno ograničenje na prve dvije slovne oznake ukoliko poznajemo sva registracijska područja u RH. One tad moraju biti u sljedećem skupu: {BJ, BM, ČK, DA, DE, DJ, DU, GS, IM, KA, KC, KR, KT, KŽ, MA, NA, NG, OG, OS, PU, PŽ, RI, SB, SK, SL, ST, ŠI, VK, VT, VU, VŽ, ZD, ZG, ŽU}.

6.2. Primjena uzoraka

Ulaz u sintaksni analizator najčešće odgovara izlazu neuronske mreže. Za svaki detektirani znak, neuronska mreža generira izlazni vektor. Na temelju tog vektora možemo izraditi sortirani skup kandidata na *i*-tom mjestu. Sintaksnom analizatoru tad proslijedimo listu svih skupova. Duljina liste će odgovarati broju pronađenih znakova.

Ponekad će se kroz algoritme segmentacije i klasifikacije „provući“ znakovi koji nisu dio tablice, već odgovaraju rubovima registarske tablice (često se zna dogoditi da sustav prepozna rub tablice kao slovo „I“). Takva greška može na razini sintaksnog analizatora uzrokovati da duljina liste znakova ne odgovara niti jednom definiranom uzorku. U takvim slučajevima je razumno odbaciti rubne znakove i pokušati provesti sintaksnu analizu s najsličnijim uzorkom.

Ipak, u većini će slučajeva duljina liste odgovarati nekom od definiranih uzoraka. Ako se na nekom mjestu pronađe znak koji ne zadovoljava sintaksne uvjete, analizator će pokušati napraviti ponovnu evaluaciju zamjenom krivog znaka s idućim kandidatom iz liste. Ukoliko niti jedan znak ne zadovolji uvjete, analizator će dojaviti da znak ne može biti prepoznat.

7. Testovi i završna razmatranja

7.1. Baza slika

Za potrebe testiranja razvijene programske potpore iskorištena je baza od 248 fotografija automobila. U skupu fotografija nisu uključene tablice za čije prepoznavanje sustav nije namijenjen. Uvjeti koje su fotografije morale zadovoljiti da bi bile uvrštene u testnu bazu su sljedeći:

- Registarska tablica mora biti vidljiva, dovoljno velika i čitljiva ljudskom oku.
- Tablica ne smije biti prikazana pod prevelikim kutom (10° i više).
- Registarska tablica mora biti iz Republike Hrvatske, no ne očekuje se da sustav prepoznaje hrvatske dijakritičke znakove.

Među fotografijama koje zadovoljavaju navedene uvjete može se napraviti podjela po težini prepoznavanja. Definirajmo sljedeće skupove:

- S_n - skup čistih tablica
- S_s - skup tablica lošeg osvjetljenja ili kontrasta
- S_k - skup kosih tablica
- S_o - skup tablica sa zahtjevnom okolinom

Tada je baza testnih fotografija skup $S = S_n \cup S_s \cup S_l \cup S_o$.

7.2. Uspješnost prepoznavanja

Postoje dva osnovna načina definiranja uspješnosti prepoznavanja tablice: binarni rezultat i težinski rezultat.

Binarni rezultat je zasnovan na jednostavnom principu: ukoliko su svi znakovi na prepoznatoj tablici jednaki onima sa stvarne tablice, tada je rezultat 1, inače 0.

$$r_b(T) = \begin{cases} 0, \text{ ako je } \mathrm{T}_{\text{prepoznato}} \neq \mathrm{T}_{\text{stvarno}} \\ 1, \text{ ako je } \mathrm{T}_{\text{prepoznato}} = T_{\text{stvarno}} \end{cases}$$

Težinski rezultat se definira kao omjer uspješno prepoznatih znakova i ukupnog broja znakova na registarskoj tablici.

$$r_t(T) = \frac{n_{prepoznatih}}{n_{ukupno}}$$

Na primjer, ako registarsku tablicu „ZD328SS“ sustav prepozna kao „ZD3285S“ tada je binarni rezultat 0, a težinski 0.86.

Ukupna uspješnost prepoznavanja se definira kao aritmetička sredina uspješnosti prepoznavanja pojedine tablice.

$$P(S) = \frac{1}{n}\sum_{i=0}^{n-1} r(P_i)$$

Konačno, tablica 4. prikazuje uspješnost prepoznavanja tablica iz baze fotografija podijeljene na podskupove. Za izračun stope uspješnosti korišteni su težinski rezultati.

Tablica 4. Uspješnost prepoznavanja

Podskup	Broj tablica	Ukupan broj znakova	Uspješnost prepoznavanja
Čiste tablice	181	1274	66.91%
Tablice s lošim osvjetljenjem	25	181	49.43%
Kose tablice	22	157	50.76%
Tablice sa zahtjevnom okolinom	20	143	30.09%

8. Zaključak

Cilj ovog rada bio je prikaz cjelokupnog programskog sustava prepoznavanja registarskih tablica, zajedno s matematičkim aspektima izloženih algoritama. Rad je podijeljen u logičke cjeline koje objedinjuju pojedine korake u procesu prepoznavanja.

Iako su dobiveni rezultati zadovoljavajući (uspješnost prepoznavanja se u slučaju „običnih“ tablica penje i preko dvije trećine), rezultati pokazuju kako je sustav osjetljiv na promjenu uvjeta okoline. Bitno je napomenuti kako su greške u ovakvom sustavu kumulativne, dakle neuspjeh ili djelomični uspjeh prvog koraka procesa prepoznavanja uzrokuje nezadovoljavajuće rezultate svih narednih algoritama. U većini slučajeva u kojima su znakovi neuspješno prepoznati radilo se o neuspjehu lokalizacije tablice. Ukoliko bi se koristio specijalizirani program za predobradu slika i ekstrakciju registarskih tablica, sustav bi imao mnogo veće šanse za uspješno prepoznavanje znakova. Daljnja mjesta za proširenje sustava su ugradnja algoritma Houghove transformacije za ispravljanje nakošenih tablica i korištenje adaptivne metode određivanja praga za binarizaciju slike.

No i bez dodatnih nadograđivanja, može se reći kako sustav uspješno iskorištava kombinaciju algoritama iz raznih matematičkih i računalnih problemskih područja u svrhu obavljanja zadanog posla. Razvijena programska potpora dostupna u prilogu ovog rada omogućuje demonstraciju opisanih principa na proizvoljnim fotografijama.

9. Literatura

[1] Lucena, R. **ANPR Tutorial**, 24. kolovoza 2006, *Automatic Number Plate Recognition Tutorial* , http://www.anpr-tutorial.com/, 8. svibnja 2008

[2] Constant, M. *CCTV Information - Automatic Number Plate Recognition (ANPR)* http://www.cctv-information.co.uk/constant3/anpr.html, 5. svibnja 2008.

[3] Refenes A.N., Zapranis A., Francis G., **Stock performance modeling using neural networks. A comparative study with regression models**, Neural Networks 72 (1994), str. 375–388.

[4] Jacobs R.A., **Increased rates of convergence through learning rate adaptation**, Neural Networks 1 (1988), str. 295–307.

[5] Kraupner K. **Uporaba višeslojnog perceptrona za raspoznavanje brojčano-slovčanih znakova na registarskim tablicama**, diplomski rad, Fakultet elektrotehnike i računarstva, 2003.

[6] Lončarić, S. **Neuronske mreže**, predavanja, Fakultet elektrotehnike i računarstva

[7] Martinsky, O. **Algorithmic and mathematical principles of automatic number plate recognition systems**, diplomski rad, Brno University of Technology, 2007.

[8] Kirillow, A. **Neural Network OCR,** 11. kolovoza 2005., http://www.codeproject.com/KB/cs/neural_network_ocr.aspx, 12. travnja 2008.

[9] Bishop, C.M. **Neural Networks for Pattern Recognition,** Aston University, 1995.

[10] Draghici, S. **A neural network based artificial vision system for licence plate recognition,** Wayne State University

[11] Kahraman F., Kurt B., Gokmen M**. License Plate Character Segmentation Based on the Gabor Transform and Vector Quantization**, Istanbul Technical University Institute of Informatics

Raspoznavanje znakova na registarskim tablicama

Sažetak

Ovaj rad se bavi problematikom automatskog prepoznavanja registarskih tablica. Izloženi su principi detekcije područja registarske tablice s proizvoljne fotografije i algoritmi segmentacije prepoznate tablice. Nakon toga je prikazan princip normalizacije dobivenih znakova te izlučivanja karakterističnih značajki. Opisan je problem klasifikacije i pristupi za njegovo rješavanje uz pomoć neuronske mreže. Uspoređuju se rezultati dobiveni pomoću razvijene programske potpore, te se opisuje utjecaj različitih parametara na kvalitetu prepoznavanja.

Ključne riječi: Registarske tablice, računalni vid, OCR, raspoznavanje uzoraka, neuronske mreže.

Recognition of number plate characters

Summary

This paper deals with aspects of automatic number plate recognition systems. Principles of number plate area recognition are shown, as well as algorithms for segmentation of detected number plate. They are followed by the analysis of normalization and feature extraction algorithms. The discussion continues with an overview of the neural network approach to general classification problem. At the end of the recognition process, characters are being processed with a syntactical analyser. Finally, we compare results obtained from the developed software, and discuss the influence of different parameters on recognition performance and quality.

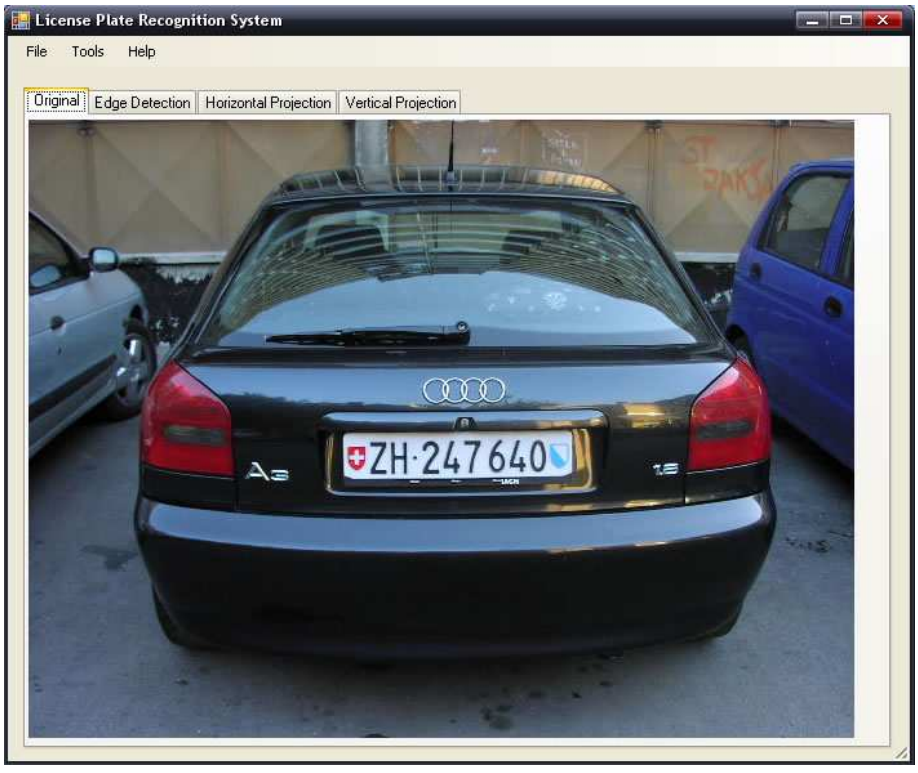
Keywords: Number plates, license plates, computer vision, OCR, pattern recognition, classification, neural networks.

Dodatak A: Programska potpora

U sklopu ovog rada razvijen je i sustav pod nazivom LPRS (*License Plate Recognition System*). Sustav je u potpunosti razvijen u programskom jeziku C#, a za svoje pokretanje zahtijeva Windows platformu na kojoj je instaliran Microsoft .NET Framework 2.0.

Dijelovi sustava se oslanjaju na AForge.NET biblioteke. AForge.NET je *framework* dizajniran za razvojne programere i istraživače iz područja računalnog vida i umjetne inteligencije. Dostupan je pod GNU GPL v2 licencom (http://code.google.com/p/aforge/). Navedene biblioteke se isporučuju zajedno s programskim sustavom u vidu DLL datoteka.

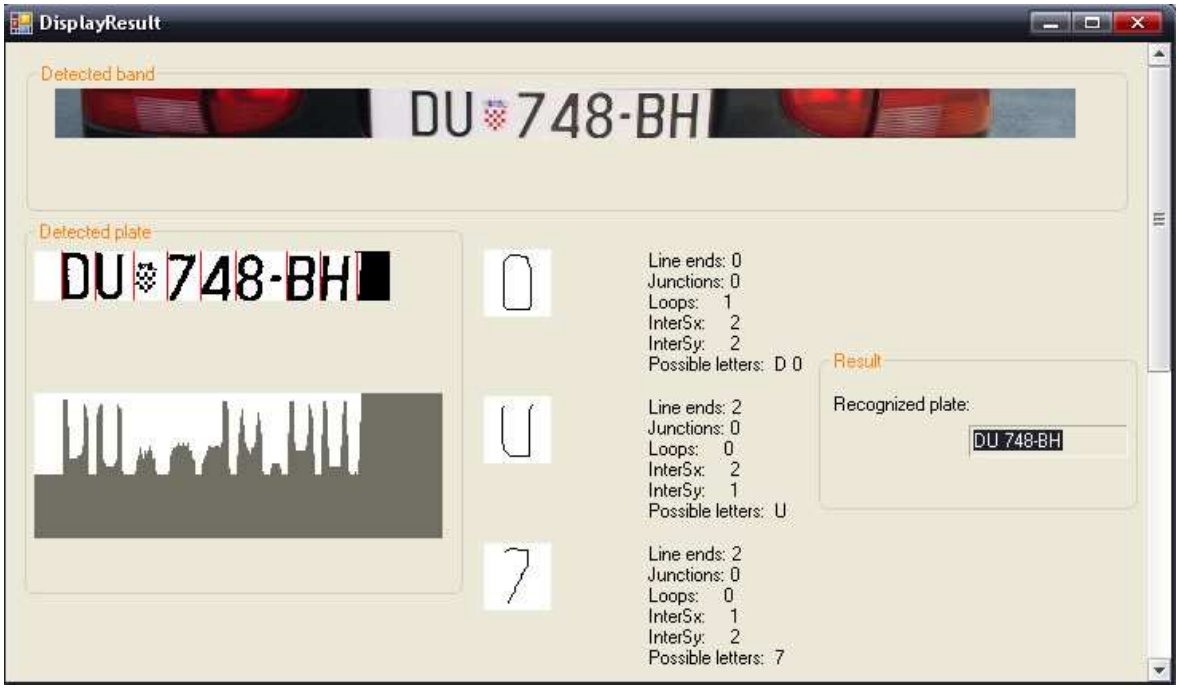
Opis sučelja programa



Slika 21. Glavno sučelje programa

Pri pokretanju programa korisnika dočekuje sučelje slično onome sa slike 21. Glavni izbornik omogućuje učitavanje slike s diska te pokretanje procesa prepoznavanja registarske tablice. Središnji dio prozora prikazuje učitanu sliku, te prikaz detekcije rubova, horizontalne i vertikalne projekcije slike.

Klikom na opciju „*Detect plate*“ iz izbornika „*Tools*“ pokreće se proces prepoznavanja tablice. Postupak je u potpunosti automatiziran. Ukoliko postoji spremljena konfiguracija neuronske mreže, korisnik će o tome biti obaviješten jednostavnom porukom. Ako neuronska mreža još ne postoji, ona će biti stvorena i spremljena na disk, a korisnika će se informirati o obavljenoj akciji.



Slika 22. Rezultat procesa prepoznavanja

Slika 22 prikazuje formu koja se prikaže nakon procesa prepoznavanja. U gornjem dijelu je prikazana prepoznata traka unutar koje se nalazi registarska tablica. S lijeve strane se nalaze detektirana tablica nakon binarizacije i njena horizontalna projekcija. Središnji dio forme zauzimaju dinamički generirane kontrole koje prikazuju sva detektirana slova nakon procesa određivanja medijalne osi, njihove karakteristične značajke, te liste mogućih znakova. Konačni rezultat programa je prikazan s desne strane, i prikazuje prepoznate znakove na tablici nakon sintaksne analize (Napomena: sintaksna analiza pokušava zadovoljiti sintaksne uzorke. Nezadovoljavajući nizovi znakova se ne odbacuju, već se prikazuju u cijelosti.)