SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br.

Eksperimentalna evaluacija metoda za prepoznavanje prometnih znakova

Andrea Žabčić

Zagreb, siječanj 2011

Tablica sadržaja

1. Uvod

S ciljem lakšeg provjeravanja ispravnosti znakova i druge prometne signalizacije razvijene su razne metode za detekciju i prepoznavanje znakova. Osnovna ideja je da se umjesto ručnog provjeravanja ispravnosti znakova koristi program koji će u snimci snimljenoj iz automobila u pokretu pronaći i prepoznati znakove, te vidjeti da li se na određenim mjestima na cesti nalaze određeni znakovi. Na taj način bi se moglo odrediti ako je neki znak nestao, previše je oštećen ili sakriven.

U ovom radu bavimo se metodama za prepoznavanje znakova. Eksperimentalno ćemo evaluirati tri često korištene metode za klasifikaciju uzoraka: stroj s potpornim vektorima, analizu svojstvenih komponenti i linearnu diskriminantnu analizu. Sve tri metode se baziraju na učenju na pripremljenom skupu slika za koje znamo u koju klasu pripadaju.

U poglavlju 2 ćemo objasniti osnovne ideje metoda za klasifikaciju uzoraka i potrebne pomoćne metode. Konkretne programske implementacije koje koristimo su objašnjene u poglavlju 3, dok se u poglavlju 4 bavimo eksperimentalnom evaluacijom danih metoda. Na kraju slijedi kratki osvrt na dobivene rezultate.

2. Prepoznavanje prometnih znakova

2.1. Pregled literature

Mnogi radovi su već objavljeni na temu ove tri metode i njihova korištenja u klasifikaciji uzoraka. U [18] je opisan jedan pristup prepoznavanju znakova korištenjem stroja s potpornim vektorima koji za računanje vektora značajki koristi histogram boje. U tom radu se opisuje algoritam za klasifikaciju slika u zadane klase.

Često se stroj s potpornim vektorima koristi prvo za klasifikaciju tipova znakova, npr. okrugli znakovi, trokutasti i drugo, i nakon toga za prepoznavanje znaka unutar skupine. U tom slučaju kao vektor značajki se prvo koristi udaljenosti do granice da bi se znak klasificirao u skupinu, i nakon toga se kao vektor značajki uzimaju vrijednosti sive slike [19] ili histogram kao u našem slučaju.

Analiza svojstvenih komponenti se koristi u [21] za prepoznavanje ljudskih lica. U tom radu se opisuje algoritam za detekciju i praćenje lica, te prepoznavanje pojedinih osoba. Analiza svojstvenih komponenti služi za projekciju uzoraka u novi prostor u kojemu su više razdvojene pojedine važne značajke lica. Te značajke ne moraju nužno biti intuitivne značajke kao što su oči, usta i slično.

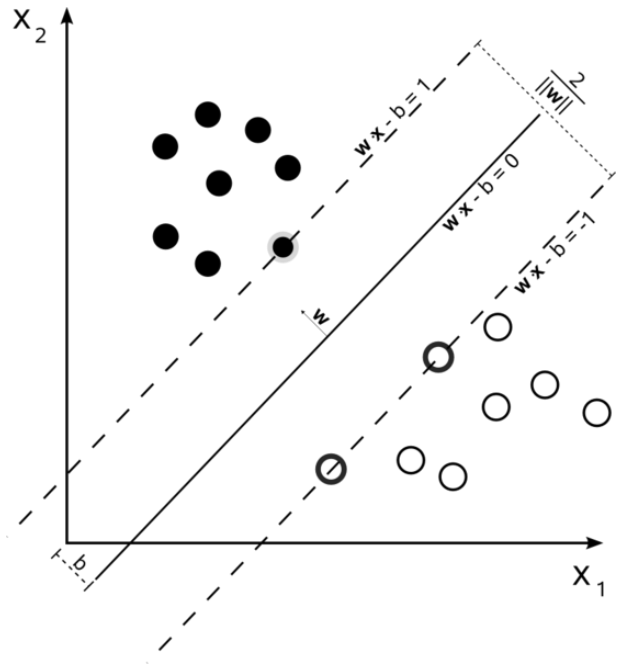
Isti problem prepoznavanja ljudskih lica se u [22] rješava korištenjem linearne diskriminantne analize. Linearna diskriminantna analiza računa prostor manje ili iste dimenzionalnosti kao originalni prostor u kojemu su uzorci koji pripadaju istoj osobi blizu jedni drugima, a uzorci različitih osoba što udaljeniji. U članku se razmatra i najbolji način odabira vektora značajki koji će najbolje razdvojiti pojedine klase.

2.2. Stroj s potpornim vektorima

Stroj s potpornim vektorima (engl. *Support Vector Machine -SVM*) je binarni klasifikator. Osnovni SVM uzima zadane ulazne podatke te ih svrstava u jednu od dvije moguće klase. Algoritam konstruira N-dimenzionalnu hiperravninu koja na optimalni način razdvaja dane klase.

Najprije ćemo promotriti jednostavan slučaj u dvodimenzionalnom prostoru, gdje su klase linearno odvojive. Podatci su prikazani kao točke u prostoru. Cilj SVM-a je pronaći

hiperravninu maksimalno udaljenu od najbližih točaka oba razreda, koja razdvaja razrede. U ovom slučaju hiperravnina je pravac. Na slici 1 je prikaz primjer linearno odvojivih skupova i pravca koji ih razdvaja.



Slika 1: optimalna hiperravnina

Podatke ćemo definirati kao skup D prikazan formulom (1).

$$D = \{(x_i, c_i) | x_i \in \mathbb{R}^p, c_i \in \{-1,1\}\}_{i=1}^{n} \quad (1)$$

Svaki podatak je određen sa dvije informacije: vektorom x_i koji je vektor značajki i oznakom c_i koja određuje klasu kojoj vekor x_i pripada.

Želimo konstruirati hiperravninu (2), gdje je **w** vektor normale pravca hiperravnine.

$$\mathbf{w} * \mathbf{x} - b = 0 \quad (2)$$

Kako bismo konstuirali optimalnu hiperravninu trebamo odabrati w i b takve da maksimiziramo marginu, odnosno udaljenosti između paralelnih hiperravnina koje su maksimalno udaljene od središnje. Te su hiperravnine definirane formulama (3) i (4).

$$\mathbf{w} * \mathbf{x} - b = 1 \quad (3)$$

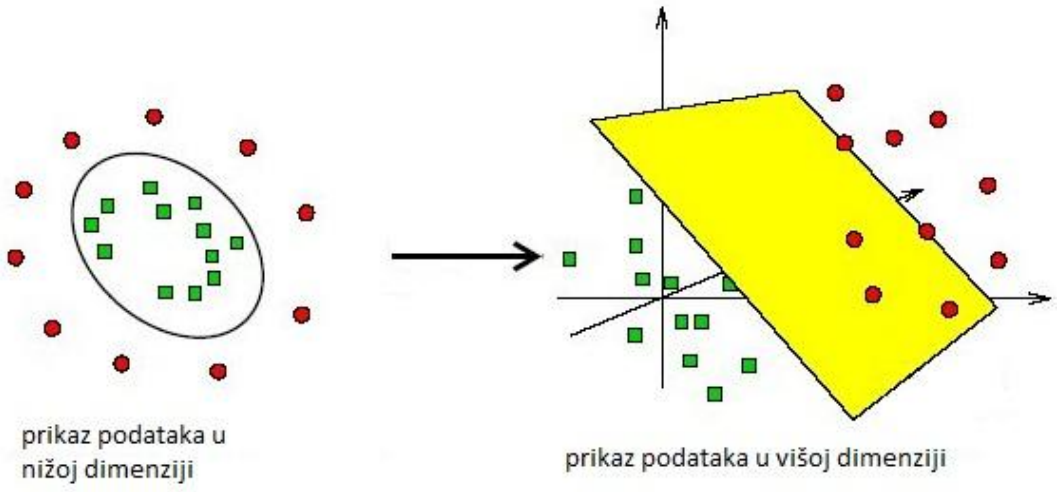
$$\mathbf{w} * \mathbf{x} - b = -1 \quad (4)$$

Bitno je napomenuti da u ovakvom modelu podatci ne smiju biti unutar margina, tj. mora vrijediti (5) i (6).

$$\mathbf{w} * \mathbf{x}_i - b \geq 1 \text{ za sve } \mathrm{x_i} \text{ prve klase} \quad (5)$$

$$\mathbf{w} * \mathbf{x}_i - b \leq -1 \text{ za sve } \mathrm{x_i} \text{ druge klase} \quad (6)$$

U realnim primjerima često podaci nisu linearno razdvojivi, te prethodno opisanim osnovnim algoritmom ne bismo mogli klasificirati podatke. U tom slučaju originalni n-dimenzionalni prostor preslikavamo u m-dimenzionalni prostor, gdje je m>n, u kojem su podatci linearno odvojivi. Taj postupak se naziva jezgreni trik (engl. *Kernel Trick*). Na slici 2 je prikazan primjer linearno neodvojivih podataka u 2-dimenzionalnom području, te preslikavanje u 3-dimenzionalni prostor u kojemu su podatci linearno odvojivi. U 3-dimenzionalnom prostoru hiperravnina više nije pravac nego 2-dimenzionalna ravnina.



Slika 2: povećanje dimenzionalnosti radi postizanja linearne odvojivosti razreda

Postupak sa jezgrenim trikom je sličan osnovnom postupku, promjena je u tome što se umjesto vektora značajki x_i koristi nelinearna funkcija ϕ (x_i). Dimenzija dobivenog prostora često zna biti puno veća od dimenzije originalnog prostora, ponekada i

beskonačna. Zbog velike dimenzionalnosti problem stvara računanje unutarnjeg produkta ϕ (x_i)***w**. Međutim, u većini slučajeva moguće je napisati jezgrenu funkciju (7), koju je moguće puno jednostavnije izračunati nego računati unutarnji produkt.

$$K(x_i, x_j) = \phi(x_i)^T * \phi(x_j) \tag{7}$$

Koristi se više različitih jezgrenih funkcija:

- Linearna: $K(x_i, x_j) = {x_i}^T * x_j$
- Polinomna: $K(x_i, x_j) = ({x_i}^T * x_j + a)^b$
- Gaussova: $K(x_i, x_j) = e^{-\gamma \|x_i - x_j\|^2}$, γ>0

Najčešće se koristi Gaussova jezgra, te ćemo ju i mi koristiti. Ova jezgra preslikava u beskonačno-dimenzionalni prostor.

SVM služi za klasifikaciju vektora značajki u dvije klase, no nama za klasifikaciju znakova treba klasifikacija između više klasa te moramo prilagoditi algoritam. Dva su moguća načina: jedan protiv jednog (engl. *One-versus-one*) i jedan protiv svih (engl. *One-versus-all*). Kod klasifikacije jedan protiv jednog za sve moguće parove klasa se grade binarni klasifikatori. Takvih klasifikatora ima $\frac{n(n-1)}{2}$, gdje je n broj klasa. Svaka klasifikacija glasa za jednu od dvije klase, te se na kraju uzorak pridjeljuje klasi sa najviše glasova. Jedan protiv svih konstruira n klasifikatora, gdje svaki klasifikator ima dvije klase: klasu tog znaka i klasu u koju spadaju svi ostali znakovi. Svaki klasifikator osim izlaza daje i mjeru sigurnosti u svoj izbor. Od svih klasifikatora koji su izabrali klasu znaka, a ne klasu svi ostali (engl. *all*), izabire se onaj koji ima najveću mjeru sigurnosti.

2.2.1. Odabir vektora značajki

Postavlja se pitanje kako postaviti vektore značajki da bi klasifikacija dala najbolje rezultate. U najjednostavnijem slučaju bi se kao vektori značajki uzele vrijednosti boja slike. Slike su standardno zapisane u RGB obliku (engl. *red, green, blue*), te bi se kao vrijednosti mogle uzeti R,G i B komponente. No takav način odabira vektora značajki nije polučio dobre rezultate u detekciji znakova. Pokazalo se da je umjesto toga bolje slike

pretvoriti u sive slike, jer je luminantna komponenta puno korisnija za raspoznavanje. Još bolje od same luminantne komponente je odabrati njezin gradijent.

Gradijent funkcije dvije varijable:

$$\nabla f(x,y) = \frac{\partial f(x,y)}{\partial x}\vec{\imath} + \frac{\partial f(x,y)}{\partial y}\vec{\jmath} \tag{8}$$

Ne možemo koristiti kontinuiranu varijantu gradijenta pošto je slika diskretna, te ćemo koristiti aproksimacije derivacija:

$$dx = \frac{\partial f(x,y)}{\partial x} = \frac{f(x+1,y) - y(x-1,y)}{2} \tag{9}$$

$$dy = \frac{\partial f(x,y)}{\partial y} = \frac{f(x,y+1) - f(x,y-1)}{2} \tag{10}$$

Iznos gradijenta računamo kao:

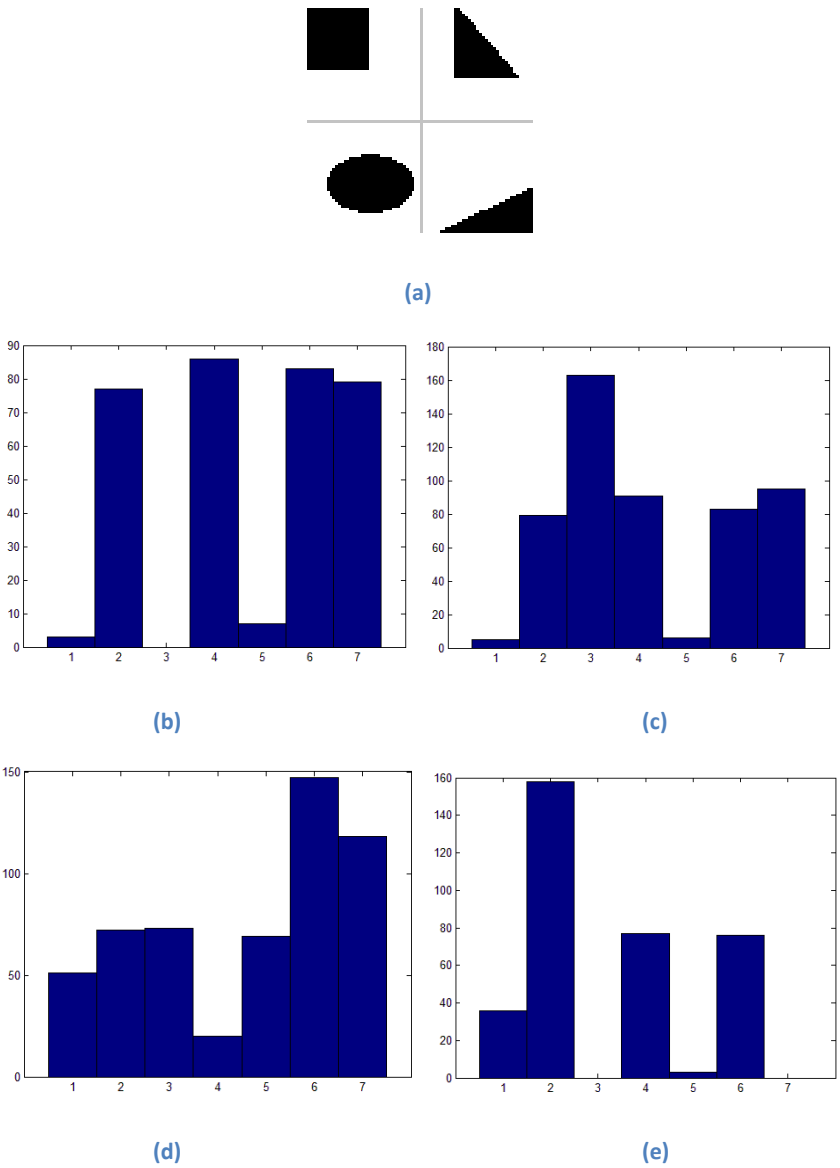
$$g = \sqrt{dx^2 + dy^2} \tag{11}$$

Orijentaciju gradijenta računamo kao:

$$\phi = arctg(\frac{dy}{dx}) \tag{12}$$

Umjesto gradijenta kao vektor značajki se koristi histogram orijentacije gradijenta. Osnovna ideja je da se slika podijeli na male segmente, te se za svaki segment računa histogram orijentacije gradijenta. Za svaki slikovni element se promatra kut gradijenta ϕ te se za segment računa histogram kutova gradijenta, koji daje informaciju o obliku koji se nalazi u segmentu. Na slici 3 je prikazana slika te histogram smjera gradijenta ukoliko ga

računamo na četiri segmenta. U ovom slučaju imamo sedam kutova raspoređenih na 360° (2π). Svaki kut je širine $\frac{2\pi}{7}$.



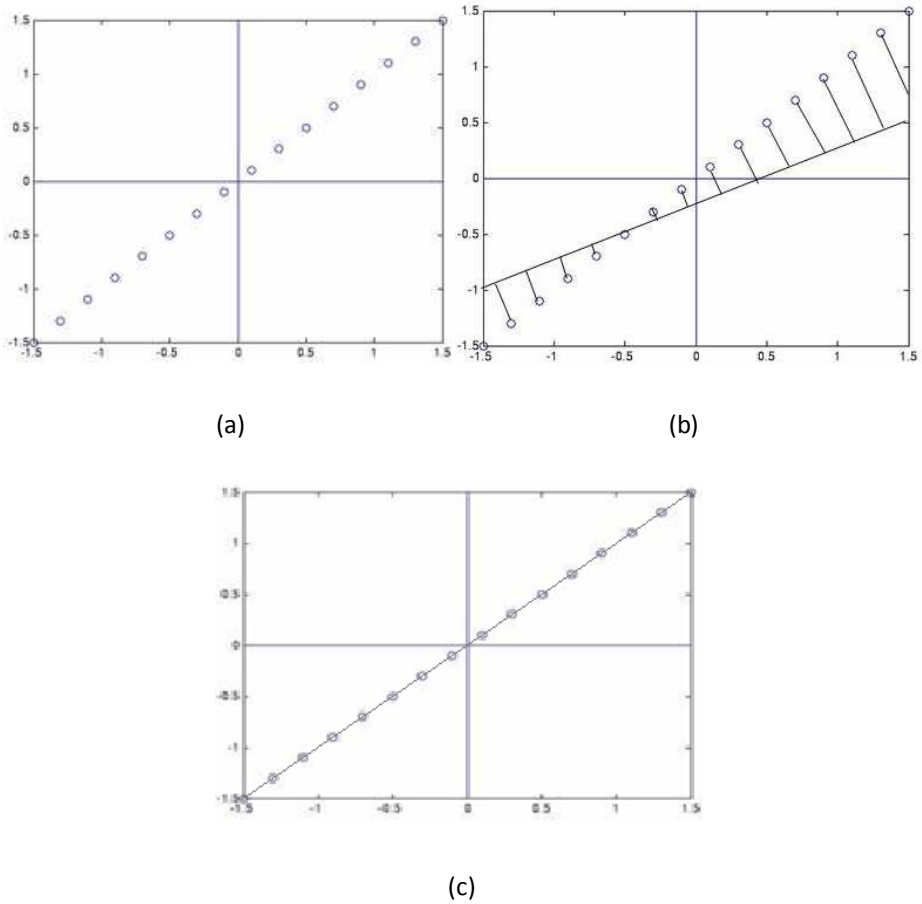
Slika 3: (a) originalna slika (b) (c) (d) i (e) histogrami orijentacije gradijenta za sedam kutova; kut 3 predstavlja kutove između $-\frac{\pi}{7}$ i $\frac{\pi}{7}$, ostali su raspoređeni u smjeru obrnutom od kazaljke na satu

2.3. Analiza svojstvenih komponenti

Analiza svojstvenih komponenti (engl. *Principal component analysis* – PCA) je metoda za analizu podataka. Ona omogućava izlučivanje najvažnijih značajki iz skupa uzoraka, i to na način da reducira dimenziju podataka bez većeg gubitka informacija.

Osnovna ideja je da se pronađe takav potprostor da pogreška između originalnog uzorka i projekcije bude što manja, te da u tom potprostoru podaci najviše variraju.

Pogledati ćemo jednostavan primjer. Podatci se nalaze u dvodimenzionalnom prostoru, a želimo ih projicirati u jednodimenzionalni prostor. Na slici 4a je prikazan ulazni skup podataka, slika 4b prikazuje jednu od mogućih projekcija, dok se na slici 4c nalazi idealna projekcija.



(a) (b)

(c)

Slika 4: (a) primjer ulaznih podataka (b) jedna od mogućih projekcija (c) najbolja projekcija

Do bismo objasnili postupak analize svojstvenih komponenti, prvo moramo definirati neke pojmove statistike i linearne algebre.

2.3.1. Kovarijacijska matrica

U statistici kovarijanca pokazuje koliko se dvije varijable mijenjaju zajedno. Uzmemo li tri skupa vrijednosti $(x_1,x_2\ldots,x_n)$, $(y_1,y_2\ldots,y_n)$ i $(z_1,z_2\ldots,z_n)$ njihove kovarijance su:

$$\sigma_{xy}^2 = \sigma_{yx}^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{n} \tag{13}$$

$$\sigma_{xz}^2 = \sigma_{zx}^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(z_i - \bar{z})}{n} \tag{14}$$

$$\sigma_{yz}^2 = \sigma_{zy}^2 = \frac{\sum_{i=1}^{n}(y_i - \bar{y})(z_i - \bar{z})}{n} \tag{15}$$

U matričnom obliku se može zapisati kao:

$$C = \begin{bmatrix} \sigma_{xx}^2 & \sigma_{xy}^2 & \sigma_{xz}^2 \\ \sigma_{yx}^2 & \sigma_{yy}^2 & \sigma_{yz}^2 \\ \sigma_{zx}^2 & \sigma_{zy}^2 & \sigma_{zz}^2 \end{bmatrix} \tag{16}$$

Takav zapis se naziva kovarijacijskom matricom.

2.3.2. Svojstveni vektori i svojstvene vrijednosti

Vektor v ≠ 0 je svojstveni vektor [10] matrice A ako za neki skalar λ zadovoljava jednadžbu:

$$\mathbf{A} \cdot \mathbf{v} = \lambda \cdot \mathbf{v} \tag{17}$$

U tom slučaju λ se naziva svojstvenom vrijednosti matrice A za zadani v. Glavno svojstvo vektora v je da se njegov smjer ne mijenja u slučaju da se on množi matricom A, nego se samo mijenja njegova magnituda.

Uz uvjet da je A kvadratna matrica jednadžba (18) se može zapisati kao:

$$(\mathbf{A} - \lambda\mathbf{I})\mathbf{v} = 0 \tag{18}$$

Da bi jednadžba imala netrivijalno rješenje (v≠0) ne smije postojati inverz:

$$(\mathbf{A} - \lambda\mathbf{I})^{-1} \tag{19}$$

Da bi taj uvjet vrijedio determinanta matrice A-λI mora biti jednaka nuli.

$$|\mathbf{A} - \lambda\mathbf{I}| = 0 \tag{20}$$

Iz te jednadžbe možemo izračunati svojstvene vrijednosti matrice A. Determinanta (21) je polinom n-tog stupnja, kojeg nazivamo karakteristični polinom matrice A i označavamo kao:

$$\kappa(\lambda) = |\mathbf{A} - \lambda\mathbf{I}| \tag{21}$$

Nakon što smo izračunali svojstvene vrijednosti, svojstvene vektore možemo izračunati tako da uvrštavamo sve vrijednosti λ u jednadžbu (18).

Kada trebamo izračunati svojstvene vektore i vrijednosti većih matrica nije više praktično to računati na osnovni način. Zato su razvijeni razni algoritmi za njihovo jednostavnije i brže računanje. Jedan od njih je Jacobijev rotacijski algoritam (engl. *Jacobi rotation algorithm*) [11].

2.3.2.1. Jacobijev rotacijski algoritam

Jacobijev rotacijski algoritam se često koristi za računanje svojstvenih vektora jer je jednostavan i stabilan, iako u slučaju većih matrica znatno sporiji od nekih drugih algoritama.

Postupak počinje od pretpostavke da se svaka kvadratna matrica A može dijagonalizirati množenjem sa matricom svojstvenih vektora V na sljedeći način:

$$Q = V^T A V \tag{22}$$

Jacobijev algoritam se bazira na nizu rotacija koje su osmišljene tako da u svakoj iteraciji podatci izvan glavne dijagonale postaju sve manji, dok matrica ne postane dijagonalna.

Postupak je sljedeći:

Inicijalizira se matrica Q(1) = A, te V(1)=I.

1. U svakoj iteraciji h se odabire element $Q_{pq}(h)$ izvan glavne dijagonale matrice Q(h) koji će se postaviti na nulu. U cikličnoj verziji algoritma uzimaju se redom svi elementi iznad glavne dijagonale.
2. Izračunava se kut rotacije.

$$\theta(h) = \frac{1}{2}\tan^{-1}\left(\frac{2Q_{pq(h)}}{Q_{qq}(h) - Q_{pp}(h)}\right) \; ako\; je\; Q_{qq}(h) \neq Q_{pp}(h) \tag{23}$$

$$\theta(h) = 90° \; inaće \tag{24}$$

3. Računa se Jacobijeva rotacijska matrica P(h):

$$P(h) = \begin{bmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & \cos(\theta(h)) & & -\sin(\theta(h)) & \\ & & & 1 & & \\ & & \sin(\theta(h)) & & \cos(\theta(h)) & \\ & & & & & \ddots \\ & & & & & & 1 \end{bmatrix} \begin{matrix} \\ \\ \leftarrow p \\ \\ \leftarrow q \\ \\ \\ \end{matrix} \tag{25}$$

$$\begin{matrix} & & \uparrow & & \uparrow & & \\ & & p & & q & & \end{matrix}$$

4. Računamo matricu Q(h+1) :

$$Q(h+1) = P(h)^T Q(h) P(h) \tag{26}$$

5. Računamo procjenu matrice V:

$$V(h+1) = V(h)P(h) \tag{27}$$

6. Ponavljamo korake od 2 do 5 dok se ne zadovolji neki zadani kriterij.

Vrijednosti na dijagonali matrice Q su svojstvene vrijednosti, a svojstveni vektori se nalaze u matrici V.

2.3.3. Analiza svojstvenih komponenti

Analizom svojstvenih komponenti želimo reducirati dimenziju podataka na način da naglasimo sličnosti i različitosti pojedinih uzoraka kako bismo kasnije lakše razdvojili pojedine klase podataka.

Za to ćemo koristiti kovarijacijsku matricu objašnjenu u poglavlju 2.3.1, te svojstvene vektore te matrice. To će nam biti korisno jer svojstveni vektori kovarijacijske matrice sa većim svojstvenim vrijednosti odgovaraju smjerovima u kojima podatci više variraju.

Prije same metode potrebno je definirati ulazni skup podataka. Vektori značajki će nam biti vrijednosti sive slike znaka određene dimenzije, prikazan kao vektor stupac veličine n. Uzmemo li N kao broj slika vektore značajki ćemo organizirati u matricu X (29) veličine n×N, gdje su vektori značajki pojedinih slika stupci matrice.

$$X = [x_1, x_2, \dots, x_N] \tag{28}$$

U slijedećom koraku trebamo od svakog vektora stupca oduzeti prosječni vektor stupac. Prosječni vektor stupac se računa kao:

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{29}$$

Oduzimanjem prosječnog vektora $\bar{x}$ od svih x_i dobivamo novu matricu A oblika:

$$A = [x_1 - \bar{x}, x_2 - \bar{x}, \dots, x_N - \bar{x}] \tag{30}$$

Zatim računamo kovarijacijsku matricu C dimenzija n×n. Nju možemo izračunati kao:

$$C = AA^T \tag{31}$$

Zatim je potrebno izračunati svojstvene vektore i vrijednosti kovarijacijske matrice. Oni se mogu računati na način opisan u poglavlju 2.3.2, no s obzirom na veličinu matrice C bolje je koristiti neki od složenijih algoritama za računanje vlastitih vektora poput Jacobijevog rotacijskog algoritma opisanog u poglavlju 2.3.2.1.

Svojstveni vektori sa većim svojstvenim vrijednostima nose više informacije a manje šuma te će u njima biti sadržane bitne značajke skupa uzoraka. Svojstveni vektori sa najmanjim svojstvenim vrijednostima samo unose šum te njih nećemo uzimati u obzir prilikom računanja nove baze prostora.

Svojstveni vektori predstavljaju novu svojstvenu bazu prostora u kojoj će se prikazivati i uspoređivati znakovi.

Svojstvene vektore ćemo spremiti po stupcima u matricu V.

$$V = [v_1, v_2, \dots, v_n] \tag{32}$$

Stupci $v_1, v_2, \dots, v_n$ su svojstveni vektori zadane dimenzije.

Tada svaku ulaznu sliku X preslikavamo u novi prostor na slijedeći način. Najprije od slike oduzmemo prosječnu sliku, te dobivamo vektor značajki x_i'. Taj vektor preslikavamo u novi prostor formulom:

$$y_i = V^T \cdot x_i' \tag{33}$$

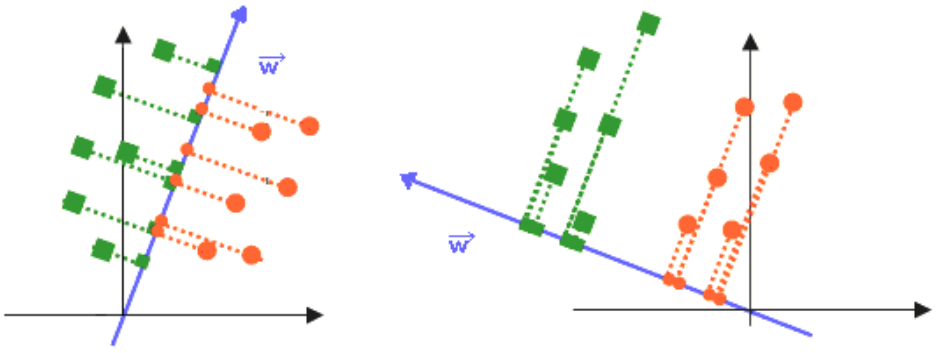
Nakon što smo preslikali podatke u zadani prostor potrebno ih je na neki način klasificirati. Klasifikacija se vrši metodom jedan najbliži susjed (engl. 1-NN – 1 *nearest neighbor*). To je specijalan slučaj metode K-najbližih susjeda (engl. K-NN – 1k *nearest neighbor*). Ta metoda klasificira uzorak na temelju najbližih uzoraka iz skupa za učenje. Pronalazi se K najbližih susjeda te se uzorak klasificira u onu klasu u kojoj se nalazi najviše susjeda. 1-NN klasificira uzorak u onu klasu u kojoj se nalazi prvi najbliži susjed. Udaljenosti između uzoraka se računa po formuli za euklidsku udaljenost. Definiramo li dva vektora značajki: $X=[x_1,x_2,\dots x_n]$ i $Y=[y_1,y_2,\dots,y_n]$ udaljenost se računa kao:

$$d(X,Y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{34}$$

2.4. Linearna diskriminantna analiza

Linearna diskriminantna analiza (engl. *Linear discriminant analysis* – LDA) je metoda koja jednako kao i analiza svojstvenih komponenti reducira dimenziju podataka, ali na način da su različite klase podataka razdvojive.

Pogledati ćemo jednostavni primjer točaka u dvodimenzionalnom prostoru koje želimo projicirat na pravac. Na slici 5 su prikazane projekcije na dva različita pravca. Projekcijom na pravac na slici 5b uzorci postaju linearno razdvojivi.



(a) (b)

Slika 5: projekcija uzoraka iz dvodimenzionalnog prostora na pravac w (a) loša projekcija – uzorci nisu linearno razdvojivi (b) dobra projekcija

Definiramo skup $X=\{x_1,x_2,\ldots,x_n\}$ kao ulazne d-dimenzionalne vektore, te skup $Y = \{y_1,y_2,\ldots,y_n\}$ koje se sastoji od projekcija skupa X na zadani vektor w. Projicirani uzroci se računaju kao:

$$y_i = \vec{w}^T \cdot x_i \tag{35}$$

Skup uzoraka X je razdvojen u C klasa, koje se moraju očuvati i u projiciranom skupu Y. Pojedinu klasu ćemo označiti kao X_i . Srednje vrijednosti pojedinih klasa ćemo računati po formuli (37) za početni skup podataka, te po formuli (38) za projicirani skup.

$$\mu_i = \frac{1}{n_i} \sum_{x \in X_i} x \tag{36}$$

$$\tilde{\mu_i} = \frac{1}{n_i} \sum_{y \in Y_i} y \tag{37}$$

Definiramo mjeru raspršenosti unutar klase kao matricu raspršenosti formulom (39), te mjeru raspršenosti između klasa formulom (40). Broj klasa je označen sa c.

$$S_W = \sum_{i=1}^{c} \sum_{x \in X_i} (x - \mu_i)(x - \mu_i)^T \tag{38}$$

$$S_B = \sum_{i=1}^{c} n_i (\mu_i - \mu)(\mu_i - \mu)^T \tag{39}$$

Može se pokazati da su uzorci najviše linearno razdvojeni ako se maksimizira raspršenost između klasa, a minimizira raspršenosti unutar klasa.

Projekcija iz d-dimenzionalnog prostora u (c-1)-dimenzionalni prostor postiže se uporabom (c-1) diskrimantnih funkcija te je dana izrazom $y_i = \vec{w_i}^T \vec{x}$, i=1,2,…c-1. Vrijednosti y_i su komponente vektora $\vec{y}$, a vektori $\vec{w_i}$ su stupci matrice W:

$$\vec{y} = W^T \vec{x} \tag{40}$$

Projicirani skup uzoraka $\vec{y}_1, \vec{y}_2, \ldots \vec{y}_n$ ima svoje srednje vrijednosti $\tilde{\mu_i}$ definirane formulom (38), te se za njih isto mogu izračunati mjere raspršenosti:

$$\tilde{S_W} = \sum_{i=1}^{c} \sum_{y \in Y_i} (y - \tilde{\mu}_i)(y - \tilde{\mu}_i)^T \tag{41}$$

$$\tilde{S}_B = \sum_{i=1}^{c} n_i(\tilde{\mu}_i - \tilde{\mu})(\tilde{\mu}_i - \tilde{\mu})^T \quad (42)$$

Može se pokazati da vrijedi sljedeći odnosi između $\tilde{S}_B$ i S_B, te $\tilde{S}_W$ i S_W:

$$\tilde{S}_W = W^T S_W W \quad (43)$$

$$\tilde{S}_B = W^T S_B W \quad (44)$$

Pošto želim maksimizirati raspršenost između klasa,a minimizirati raspršenost unutar klase, trebamo naći takav W da se maksimizira izraz:

$$J(W) = \frac{|\tilde{S}_B|}{|\tilde{S}_W|} = \frac{|W^T S_B W|}{|W^T S_W W|} \quad (45)$$

Stupci matrice W su generalizirani svojstveni vektori koji odgovaraju najvećim svojstvenim vrijednostima u:

$$S_B w_i = \lambda_i S_W w_i \quad (46)$$

Kada je matrica S_W nesingularna problem se može jednostavno riješiti. Problem nastaje kada je S_W singularna. To se često događa kada je dimenzionalnost vektora značajki puno veća od broja vektora. Tada se problem rješava tako da se sustav svede na nižu dimenzionalnost korištenjem PCA algoritma i nakon toga se primjeni LDA.

Kada imamo izračunatu matricu W svi podaci se projiciraju u novu dimenziju uvrštavanjem u formulu:

$$\vec{y} = W^T \vec{x} \quad (47)$$

Za izračun jednadžbe (47) koristiti ćemo Choleskyjevu dekompoziciju (engl. Cholesky decomposition) [12].

2.4.1. Choleskyjeva dekompozicija

Choleskyjeva dekompozicija je dekompozicija simetrične pozitivno definitne matrice u produkt donje trokutaste matrice i njene hermitske konjugirane matrice.

Matrica A je simetrična ako je realna i vrijedi $A^T = A$.

Matrica A je pozitivno definitna ako je $v^T Av > 0$ za sve realne vektore v≠0.

Kvadratna matrica A je donja trokutasta ako je $a_{ij} = 0$ za i<j.

Matricu $A^* = \bar{A}^T$ zovemo hermitskom konjugiranom matricom matrice A.

Choleskyjeva dekompozicija se može zapisati kao (48), gdje je L donja trokutasta matrica sa strogo pozitivnim vrijednostima na dijagonali.

$$A = LL^* \tag{48}$$

Choleskyjeva dekompozicija se uglavnom koristi za rješavanje jednadžbi oblika $\mathbf{Ax} = \mathbf{b}$. Ako je A simetrična i pozitivno definitna tada možemo riješiti tu jednadžbu dekompozicijom matrice $\mathbf{A} = \mathbf{LL}^{\mathrm{T}}$ te računanjem prvo jednadžbe $\mathbf{LY} = \mathbf{b}$ i nakon toga jednadžbe $\mathbf{L}^{\mathrm{T}}\mathbf{x} = \mathbf{y}$.

Choleskyjev algoritam je rekurzivan algoritam. U prvom koraku i=1 je:

$$A^{(1)} = A \tag{49}$$

U i-tom koraku matrica A poprima vrijednost:

$$A^{(i)} = \begin{bmatrix} I_{i-1} & 0 & 0 \\ 0 & a_{i,i} & b_i^* \\ 0 & b_i & B^{(i)} \end{bmatrix} \tag{50}$$

Sada možemo definirati:

$$L_i = \begin{bmatrix} I_{i-1} & 0 & 0 \\ 0 & \sqrt{a_{ii}} & 0 \\ 0 & \frac{1}{\sqrt{a_{ii}}} b_i & I_{n-1} \end{bmatrix} \tag{51}$$

Možemo zapisati $A^{(i)}$ kao:

$$A^{(i)} = L_i A^{(i+1)} L_i{}^* \tag{52}$$

Gdje je:

$$A^{(i+1)} = \begin{bmatrix} I_{i-1} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -\dfrac{1}{a_{i,i}} b_i b_i{}^* \end{bmatrix} \tag{53}$$

Ovaj postupak ponavljamo za i=1,…, n, dok ne postane $A^{(i+1)} = I$. Donja trokutasta matrica L koju tražimo se računa kao:

$$L = L_1 L_2 \dots L_n \tag{54}$$

3. Implementacija

3.1. Priprema slika

Prije samih algoritama potrebno je pripremiti slike znakova. Početne slike su snimljene iz vozila u pokretu nalik primjeru na slici 6.

Slika 6: primjer ulazne slike snimljen iz vozila u pokretu

Slike su prethodno izdvojene iz snimke te je svaka slika dobila naziv po znaku koji se nalazi na njoj. Tako je slika znaka prikazana slikom 6 dobila naziv A01_0120. Oznaka A01 govori o kojoj vrsti znaka se radi, dok 0120 označava da se radi od 120 po redu znaku A01.

U odgovarajućoj datoteci se nalaze koordinatne pozicije i širina i visina znaka na slici u sljedećem obliku:

A01_0120.bmp 1 600 168 84 80

Broj 1 iza naziva slike označava da se na ovoj slici nalazi jedan znak (uzeli smo u obzir samo trokutaste znakove, pa je ograničenje brzine zanemareno). Iza toga slijede x i y koordinate početka znaka, tj. gornji lijevi kut pravokutnika koji okružuje znak. Koordinatni sustav započinje od gornjeg lijevog ugla. Zadnja dva broja označavaju širinu i visinu pravokutnika. Na slici 7 je prikazan izdvojen znak.

Slika 7: izdvojeni znak

Korištena kamera pri snimanju koristi preplitanje (engl. *interlacing*), način snimanja gdje se prvo snimi svaka druga linija i nakon toga preostale. Rezultat toga je pojava smetnji prilikom brzih pokreta kamere. Najjednostavniji način rješavanja ovog problema je odbacivanje polovice linija i interpolacija (engl. *deinterlacing*). Na taj način od jedne slike nastaju dvije, jedna odbacivanje parnih linija, druga odbacivanje neparnih. Slika 8 prikazuje primjer slika nastalih odbacivanjem linija.

(a) (b) (c)

Slika 8: (a) originalna slika (b)slika nastala odbacivanjem parnih linija (c) slika nastala odbacivanjem neparnih linija

Za izdvajanje dijelova slike i odbacivanje linija i interpolaciju koristili smo programski alat ImageMagic [16]. ImageMagic je alat za manipulaciju slikama u različitim formatima. Verzija koju smo mi koristili se pokreće preko komandne linije. Koristili smo naredbu covert koja služi za promjenu formata slike, ali i za promjenu dimenzije slike, izrezivanje dijela slike i drugo.

Za izrezivanje samoga znaka smo koristili naredbu convert na sljedeći način:

```
convert -depth 8 -extract widthxheight+x+y sourceImage destImage
```

Opcija `-extract` određuje da će se iz slike izrezati zadani dio. Širinu i visinu izrezanog dijela određujemo upisom brojeva na mjesto `width` i `height`, a x i y koordinatu

početka pravokutnika kojega želimo izrezati upisujemo na mjesto `x` i `y`. Izvorna slika se nalazi na lokaciji `sourceImage` a rezultat će se spremiti na `destImage`.

Za *deinterlacing* koristimo naredbu `convert` na sljedeći način:

```
Convert –sample 100%x50% -resize 100%x200% sourceImage destImage

Convert –roll +0-1 –sample 100%x50% -resize 100%x200% sourceImage
destImage
```

Opcija `–sample 100%x50%` definira da će se iz originalne slike uzeti sve vrijednosti po x osi (100%) i svaka druga po y osi (50%). Opcija `-resize` ovako izdvojenu sliku uvećava po y komponenti na duplo veću sliku, odnosno na dimenzije koje je slika imala na početku. Prilikom stvaranja druge slike koristi se opcija `–roll`, čija je sintaksa `–roll{+-}x[+-}y`, koja će sliku pomaknuti za jedan redak prema gore kako bi se izdvojili parni redovi.

Za algoritme PCA i LDA je još dodatno potrebno slike pretvoriti u sive slike te ih skalirati na dimenzije 64x64. Program koji će računati histograme gradijenta za SVM će sam pretvoriti slike u sive slike te skalirati na dimenzije 48x48.

Korištenjem alata ImageMagic sliku ćemo pretvoriti u sivu te promijeniti dimenzije na sljedeći način:

```
convert –crop widthxheight+xoffset+yoffset –resize +64x+64 -
colorspace Gray –depth 8 sourceImage destImage
```

Opcija `–crop` će izrezati pravokutni dio slike širine `width` i visine `height`, pomaknuti od početka slike za `xoffset` u smjeru x i `yoffset` u smjeru y. Opcija `–resize` će slici promijeniti dimenziju na zadanu, dok će opcija `-colorspace Gray` sliku pretvoriti u sivu sliku. Na slici 9 su prikazani primjeri sivih slika.

Slika 9: sive slike

3.2. Stroj s potpornim vektorima

Za pripremu slika se koristi biblioteka OpenCV [14]. Prije računanja vektora značajki potrebno je promijeniti dimenzije slike na 48x48, pretvoriti sliku u sivu i normalizirati vrijednosti slike.

Dimenzije slike se mijenjaju korištenjem funkcije:

```
void cvResize( const CvArr* I, CvArr* J, int
                        interpolation=CV_INTER_LINEAR );
```

Ta funkcija mijenja dimenzije izvorne slike `I` i sprema promijenjenu sliku u odredišnu sliku `J`. Dimenzije slike su određene dimenzijama odredišne slike.

Sliku pretvaramo u sivu sliku korištenjem funkcije:

```
void cvCvtColor( const CvArr* src, CvArr* dst, int code );
```

Parametar `code` postavljamo na `CV_BGR2GRAY` što označava da sliku `src` formata BGR transformira u sliku formata GRAY i sprema ju na poziciju `dst`.

Na slici 10 je prikazana siva slika nastala korištenjem funkcije `cvCvtColor`.

Slika 10: siva slika

Zatim ćemo normalizirati vrijednosti slike tako da nam vrijednosti budu između 0 i 255, i na kraju ćemo filtrirati slike Gaussovim filtrom:

$$f = \frac{1}{15}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 3 & 2 \\ 1 & 2 & 1 \end{bmatrix} \tag{55}$$

Filtriranje Gaussovim filtrom ćemo napraviti korištenjem funkcije OpenCva:

```
void cvSmooth( const CvArr* src, CvArr* dst,
                        int smoothtype=CV_GAUSSIAN,
                        int param1=3, int param2=0 );
```

Takvim filtriranjem postižemo uglađivanje rubova slike i smjer gradijenta postaje ravnomjerno raspoređen na rubovima slike.

Na slici 11 su prikazane slike nakon normalizacije vrijednosti i izglađivanja vrijednosti.

Slika 11: normalizirana i izglađena slika

3.2.1. Histogram orijentacije gradijenta

Za računanje histograma orijentacije gradijenta koristimo implementaciju objašnjenu u [1].

Prije računanja histograma orijentacije gradijenta moramo izračunati gradijente u oba smjera. To ćemo raditi korištenjem funkcije OpenCVa cvSobel:

```
void cvSobel( const CvArr* I, CvArr* J, int dx,
                        int dy, int apertureSize=3 );
```

Način na koji smo mi pozivali zadanu funkciju:

```
cvSobel(img,dx,1,0,CV_SCHARR);
cvSobel(img,dy,0,1,CV_SCHARR);
```

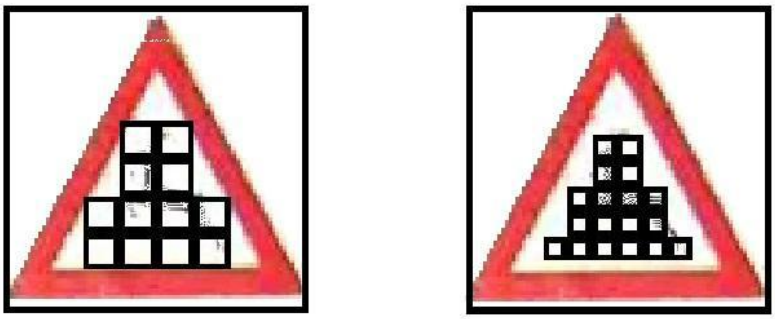
Parametar `CV_SCHARR` označava da ćemo umjesto Sobela koristiti 3x3 Scharrov filtar koji daje preciznije rezultate. U slikama dx i dy se nalaze odgovarajući gradijenti, koje možemo vidjeti na slici 12.

Slika 12: gradijenti dx i dy

Scharrov filtar daje precizniji gradijent jer uzima u obzir i dijagonalne vrijednosti. Scharrov operator je definiran maskama:

$$G_x = \begin{bmatrix} 3 & 10 & 3 \\ 0 & 0 & 0 \\ -3 & -10 & -3 \end{bmatrix} \quad G_y = \begin{bmatrix} 3 & 0 & -3 \\ 10 & 0 & -10 \\ 3 & 0 & -3 \end{bmatrix} \tag{56}$$

Pojedine klase trokutastih znakova se razlikuju samo u oblicima koji se nalaze u sredini znaka, te ćemo tu informaciju iskoristiti prilikom odabira relevantnog dijela slike. Histogrami orijentacije gradijenta se računaju nad blokovima smještenima u središtu slike. Koristit ćemo dva skupa blokova, jedne veličine 4x4 i druge veličine 6x6. Slika 13 prikazuje raspored blokova.



Slika 13: raspored blokova [1]

Histogrami nad blokovima veličine 6x6 se rade za četiri kuta i 180 stupnjeva, a histogrami nad blokovima veličine 4x4 se rade za sedam kutova i 360 stupnjeva.

Konačni histogram se dobiva tako da se prvo izračuna histogram za blokove veličine 4x4 i nakon toga za blokove 6x6, te se te vrijednosti tim redoslijedom stavljaju u vektor.

Pogledati ćemo kako se računaju histogrami orijentacije gradijenta sa blokovima veličine 4x4 i sedam kutova u 360 stupnjeva. Odluka da će se koristiti sedam kutova znači da će se cijeli skup kutova od 360 stupnjeva, odnosno 2π radijana, podijeliti na točno 7 skupina. Širina jedne skupine kutova je tada $\frac{2\pi}{7}$. Kut od nula stupnjeva će se nalaziti na sredini. Pošto se radi o neparnom broju skupina on će biti u trećoj skupini. To znači da će

svi kutovi od $-\frac{\pi}{7}$ do $\frac{\pi}{7}$ pripadati u treću skupinu te će u histogramu orijentacije gradijenta činiti jedan stupac. Na taj način računamo histograme gradijenta za sve blokove.

Slično će vrijediti i za blokove veličine 6x6 sa 180 stupnjeva samo će tamo kutovi biti raspoređeni u 180 stupnjeva a ne svih 360, što će značiti da će se negativni kutovi preslikati u pozitivne te će smjer gradijenta biti nevažan.

3.2.2. Stroj s potpornim vektorima

Za klasifikaciju strojem s potpornim vektorima ćemo koristiti biblioteku libSVM (engl. *A Library for Support Vector Machine*) [15]. Biblioteka je implementirana u programskim jezicima C++, Java, Python i Matlab, a mi ćemo koristiti implementaciju u C++.

Osim samog treniranja i testiranja biblioteka omogućava skaliranje ulaznih podataka na vrijednosti u rasponu [-1,1]. To ćemo obaviti pozivom programa:

```
svm-scale ulaznaDatoteka > izlaznaDatoteka
```

Biblioteka nudi mogućnost optimiranja parametara C i γ za Gaussovu jezgru. To je moguće napraviti korištenjem skripte *grid.py* koja optimira parametre postupkom unakrsne validacije.

Nakon odabira optimalnih parametara možemo trenirati stroj skaliranim ulaznim podacima. To obavljamo pozivom programa:

```
svm-train -c 32 -g 0.0078125 ulaznaDatoteka
```

Program će u slučaju uspješnog izvršavanja u istome direktoriju stvoriti datoteku ulaznaDatoteka.model koja sadrži parametre modela.

Klasifikacija uzoraka se vrši pozivom programa:

```
svm-predict testDatoteka model izlaznaDatoteka
```

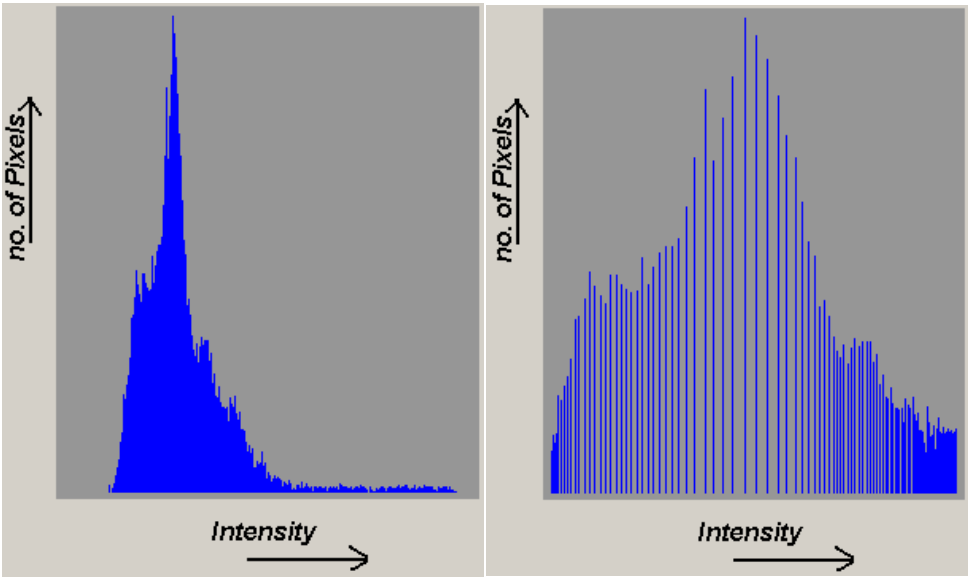
U izlaznoj datoteci će se nalaziti redom za sve testne uzorke indeksi klase u koje su klasificirani uzorci.

3.3. Analiza svojstvenih komponenti

Sa slika pripremljenih kako je opisano u poglavlju 3.1. se nakon učitavanja u program uklanja pozadina korištenjem maske, te na taj način izdvajamo samo područje interesa. Na slici 15 je prikazan rezultat primjene maske na ulaznoj slici.

Slika 14: primjena maske na ulaznoj slici

Nakon izdvajanja područja interesa još moramo izjednačiti histogram slike. Histogram slike je grafički prikaz distribucije vrijednosti slike. On pokazuje broj slikovnih elemenata za svaku vrijednost. Pošto su naše slike sive imamo samo jednu vrijednost za svaki piksel – svjetlinu. Ona se nalazi u intervalu [0,255]. Izjednačavanje histograma služi za promjenu svjetline i kontrasta slike, na željeni način. Ideja je da se promjenom srednje vrijednosti svjetline slike mijenja osvjetljenje, a promjenom standardne devijacije svjetline mijenja kontrast. Na slici 15 je prikazan jedan primjer izjednačavanja histograma.



Slika 15: (a) originalni histogram (b) izjednačen histogram

Ovaj korak je potrebno provesti jer različite slike u bazi imaju različita osvjetljenja, te bi ih htjeli normalizirati. Na slici 16 je prikazan rezultat izjednačavanja histograma slike

znaka. Usporedimo li ju sa slikom 14 možemo vidjeti da nam je izjednačavanje histograma povećalo kontrast te istaknulo traženi znak.

Slika 16: rezultat izjednačavanja histograma

Ulazni podatci su nam organizirani kako je objašnjeno u poglavlju 2.3.3. u matricu A veličine nxN gdje je n veličina vektora značajki a N broj slika. Kovarijancijska matrica se računa po formuli 58 te je dimenzija nxn.

$$C = AA^T \tag{57}$$

U sljedećem koraku potrebno je izračunati svojstvene vektore matrice C što je dugotrajan proces. Zato u slučaju da je n>N je bolje računati svojstvene vektore matrice C' (59) koja je tada dimenzija NxN.

$$C' = A^T A \tag{58}$$

Za matricu C' svojstveni vektori zadovoljavaju jednadžbu:

$$A^T A v_i = \lambda_i v_i \tag{59}$$

Množimo jednadžbu s lijeva matricom A:

$$AA^T A v_i = \lambda_i A v_i \tag{60}$$

Uzmemo li u obzir definiciju svojstvenih vektora možemo zaključiti da su Av_i svojstveni vektori matrice AA^T. Također se može pokazati da su svojstvene vrijednosti odgovarajućih svojstvenih vektora jednake za obje matrice.

Iz jednadžbe 61 možemo definirati odnos svojstvenih vektora matrica C i C':

$$\lambda_i = \lambda_i' \; i = 1, \dots, N \tag{61}$$

$$v_i = \frac{1}{\sqrt{\lambda_i}} A v_i \; i = 1, \dots, N \tag{62}$$

3.3.1. Izračun svojstvenih vektora i svojstvenih vrijednosti

U poglavlju 2.3.2.1 smo opisali Jacobijevu rotacijsku metodu za izračun svojstvenih vektora koju ćemo koristiti za izračun svojstvenih vektora matrice C. Zbog složenosti opisanog postupka mi ga nećemo koristiti na opisani način, nego ćemo koristiti procjene sinusa i kosinusa, te ćemo umjesto računanja matrice Q u novom koraku preko množenja matrica (26) definirati jednostavnije izračune. Sama implementacija Jacobijeve rotacijske metode je preuzeta iz OpenCV-a.

Prije početka incijaliziramo matricu V na jediničnu matricu dimenzija jednakih dimenzijama matrice C, te nam je početna matrica Q jednaka ulaznoj matrici C.

Zatim mijenjamo matrice Q i V određeni broj puta na sljedeći način. Uzimamo redom sve vrijednosti p i q:

$$p = 1, \dots, n-1 \tag{63}$$

$$q = p + 1, \dots, n \tag{64}$$

Računamo kosinus i sinus:

$$sin\theta = \frac{x}{\sqrt{2(1+\sqrt{1-x^2})}} \tag{65}$$

$$x = \frac{-Q_{pq}}{\sqrt{{Q_{pq}}^2 + y^2}} \; ako\ je\ y > 0 \tag{66}$$

$$x = \frac{Q_{pq}}{\sqrt{{Q_{pq}}^2 + y^2}} \; ako\ je\ y < 0 \tag{67}$$

$$y = \frac{1}{2}(Q_{pp} - Q_{qq}) \tag{68}$$

$$cos\theta = \sqrt{1 - sin\theta^2} \tag{69}$$

Računamo vrijednosti nove matrice Q na sljedeći način:

$i = 1, \dots, p-1$

$$Q_{ip} = Q_{ip}cos\theta - Q_{iq}sin\theta \quad (70)$$

$$Q_{iq} = Q_{iq}cos\theta + Q_{ip}sin\theta \quad (71)$$

$i = p, \dots, q-1$

$$Q_{pi} = Q_{pi}cos\theta - Q_{iq}sin\theta \quad (72)$$

$$Q_{iq} = Q_{iq}cos\theta + Q_{pi}sin\theta \quad (73)$$

$i = q, \dots, n$

$$Q_{pi} = Q_{pi}cos\theta - Q_{qi}sin\theta \quad (74)$$

$$Q_{qi} = Q_{qi}cos\theta + Q_{pi}sin\theta \quad (75)$$

$$Q_{pp} = Q_{pp}cos^2\theta + Q_{qq}sin^2\theta - a \quad (76)$$

$$Q_{qq} = Q_{pp}sin^2\theta + Q_{qq}cos^2\theta + a \quad (77)$$

$$a = 2Q_{pq}cos\theta sin\theta \quad (78)$$

$$Q_{pq} = Q_{qp} = 0 \quad (79)$$

Prethodno opisani algoritam se ponavlja dok matrica Q ne bude dovoljno dijagonalizirana. Uvjet zaustavljanja ćemo definirati korištenjem parametra ε prethodno postavljenog na neku malu vrijednosti. U našem slučaju ε je postavljen na e^{-15}. Definiramo:

$$Q_{norm} = \sqrt{\sum_{i=0}^{n-1}\sum_{j=0}^{i-1} {Q_{ij}}^2} \quad (80)$$

$$Q_x = \frac{Q_{norm}}{n}\varepsilon \quad (81)$$

$$Q_{max} = Q_{norm} \quad (82)$$

Postupak ponavljamo dok je zadovoljen uvjet:

$$Q_{max} > Q_x \quad (83)$$

U svakom koraku se vrijednost Q_{max} mijenja na sljedeći način:

$$Q_{max} = \frac{Q_{max}}{n} \tag{84}$$

Na kraju je još samo potrebno sortirati svojstvene vektore po pripadajućim svojstvenim vrijednostima, jer vektori sa većim svojstvenim vrijednostima nose više informacije.

3.3.2. Projekcija podataka u nižu dimenziju

Izračunati svojstveni vektori sa pripadajućim svojstvenim vrijednostima predstavljaju novu bazu prostora. Osim njih potrebno je još zapamtiti i prosječni vektor značajki, te dimenzije početnog i konačnog sustava. Dimenzija originalnog sustava je n, gdje n predstavlja veličinu vektora značajki. Dimenzija konačnog sustava će ovisiti o odnosu vrijednosti n i N, koji predstavlja broj uzoraka. Ukoliko je n>N dimenzija sustava će biti N, a ako je n<N dimenzija će biti n.

Potrebno je definirati način projekcije originalnog vektora značajki u projicirani prostor. Ukoliko je n dimenzija početnog prostora, m dimenzija projiciranog prostora, $\bar{X}$ prosječni vektor značajki početnog prostora i V svojstveni vektori možemo definirati formulu projekcije:

$$X'_i = \sum_{j=0}^{n-1} (X_j - \bar{X}_j) V_{ij} \;\; za\; i = 0, \dots, m-1 \tag{85}$$

3.3.3. Klasifikacija korištenjem pravila metode 1-NN

Nakon projiciranja podataka u zadani prostor određen svojstvenim vektorima kovarijacijske matrice potrebno je definirati način klasifikacije uzoraka. Za to će se koristiti metoda 1-NN opisana u poglavlju 2.3.3.

Svaki ispitni vektor značajki će se projicirati u zadani prostor te će se klasificirati u klasu u kojoj se nalazi najbliži susjed. Udaljenost između pojedinih vektora značajki se računa formulom euklidske udaljenosti.

3.4. Linearna diskriminantna analiza

Slike znakova se prije provođenja algoritma LDA obrađuju na jednaki način kao i kod algoritma PCA: izdvaja se područje interesa i izjednačava se histogram.

Iako postoji mogućnost direktnog korištenja LDA na skupu za testiranje, mi smo prvo korištenjem PCA smanjili dimenziju prostora sa 4096 (46x46) na 75 te nakon toga proveli postupak LDA na tom prostoru.

Nakon što smo provedbom PCA dobili novi potprostor i projicirali sve uzorke za treniranje u zadani prostor potrebno je izračunati mjeru raspršenosti unutar klasa i između klasa kako je opisano u poglavlju 2.4.

Jednom kada smo izračunali matrice raspršenost između razreda S_B i unutar razreda S_W potrebno je rješavanjem jednadžbe (87) izračunati svojstvene vektore W.

$$S_B w_i = \lambda_i S_W w_i \tag{86}$$

Jednadžbu ćemo riješiti korištenjem Choleskyjeve dekompozicije opisane u poglavlju 2.4.1.

Za izračun jednadžbe (87) ćemo koristiti biblioteku Lapack koja služi za rješavanje problema linearne algebre kao što su rješavanje linearnih sustava, računanje svojstvenih vrijednosti i vektora, dekompozicije matrica i slično.

Koristimo funkciju biblioteke čiji je prototip:

```
void dsygv_(int itype, char jobz, char uplo,  int  n,  double
       *a,  int  lda,  double *b, int ldb, double *w, int
       *info);
```

Ona računa sve svojstvene vrijednosti i svojstvene vektore realnog simetrično-definitnog problema svojstvenih vektora u obliku Ax=λBx, ABx=λ ili BAx=λx. U našem

slučaju to je prvi oblik te se upravo to odabire prvim parametrom `itype` . Parametrom `jobz` određujemo da li ćemo računati samo svojstvene vrijednosti ili i svojstvene vektore. Parametar `uplo` određuje da li će se spremiti donji trokut matrica A i B ili gornji. Parametar n određuje redoslijed matrica A i B. Zatim dolazi matrica A i njezina dimenzija `lda`, te matrica B i njezina dimenzija `ldb`. Na izlazu će matrica A sadržati matricu svojstvenih vektora. Matrica B će na izlazu predstavljani trokutastu gornju ili donju matricu iz Choleskyjeve dekompozicije. U nizu W će se na izlazu programa nalaziti svojstvene vrijednosti u uzlaznom redoslijedu.

Ovako izračunati svojstveni vektori nisu konačni jer se mora uzeti u obzir i prethodno izračunavanje PCA. Svojstveni vektori se računaju kao umnožak svojstvenih vektora LDA postupka i svojstvenih vektora izračunatih prilikom računanja PCA. Konačni svojstveni vektori su dimenzije N_{PCA}xn, gdje je N_{PCA} dimenzija u koju se projiciraju uzorci postupkom PCA, a n početna dimenzija sustava. Konačne svojstvene vrijednosti su svojstvene vrijednosti izračunate računanjem LDA potprostora, a srednja vrijednost je srednja vrijednost PCA postupka.

Podatci se projiciraju u novi prostor na jednak način kao i kod PCA postupka što je objašnjeno u poglavlju 3.3.2. Klasifikacija se također obavlja na jednak način korištenjem 1NN metode objašnjene u poglavlju 3.3.3.

4. Evaluacija

4.1. Ulazne slike

Ulazne slike su nam podijeljene u dvije skupine: T2009, koja se sastoji od 2092 slike znakova, koja nam je služila za treniranje algoritama i T2010 koja je služila za testiranje. Na slici 17 je prikazano nekoliko promjera ulaznih slika skupine T2009. Skup za učenje se sastoji od 26 klasa znakova.

Slika 17: primjer slika skupa za treniranje

Skup za testiranje sastoji se od dvije vrste slika. Prva skupina se sastoji od slika na kojima se nalaze znakovi koji su ručno označeni, dok se druga skupina sastoji od detekcija znakova koji često nisu centrirani. Na slici 18 je prikazan primjer označenog znaka i detektiranog.

Slika 18: (a) označeni znak (b) detektirani znak

Na slici se vidi da se radi o istom znaku no algoritam za detekciju nije savršeno izdvojio znak kao što je to napravljeno na označenoj slici.

4.2. Učenje na centriranim znakovima

Proveli smo treniranje sva tri algoritma na osnovnom skupu za treniranje T2009, a testirali smo sa anotacijama i detekcijama.

Za sve tri skupine je postotak točne klasifikacije nešto veći za označene znakove u odnosu na detektirane, što je bilo i očekivano pošto se u skupu za treniranje i nalaze označeni znakovi.

Postoci točne klasifikacije znakova korištenjem algoritma SVM su prikazani u tablici 1.

Tablica 1: točnost klasifikacije znakova algoritmom SVM

	Točnost klasifikacije (%)
Označeni znakovi	95.4936
Detektirani znakovi	84.4587

Kod klasifikacije algoritmom PCA znatnu ulogu u točnosti ima izabrana dimenzija novog prostora, točnije broj svojstvenih vektora koje ćemo uzeti. Već smo spomenuli da je većina informacija pohranjena u svojstvenim vektorima sa većim svojstvenim vrijednostima dok oni sa manjim svojstvenim vrijednostima samo unose grešku. Pogledati ćemo kako broj svojstvenih vektora utječe na točnost klasifikacije detektiranih znakova.

Tablica 2: točnosti klasifikacije detektiranih znakova algoritmom PCA

Broj vektora	Točnost klasifikacije (%)
10	74.5182
20	77.3019
30	78.6938
40	77.8373
50	77.3019
60	77.3019
120	76.9807

Kao što možemo vidjeti u tablici 2 nije dobro uzeti niti premalo niti previše svojstvenih vektora. U ovom slučaju je najoptimalnije uzeti oko 30-40 svojstvenih vektora iako to nije uvijek tako. Za označene znakove je točnost ovog algoritma nešto bolja, što možemo vidjeti u tablici 3.

Tablica 3: točnost klasifikacije označenih znakova algoritmom PCA

Broj vektora	Točnost klasifikacije (%)
10	85.6531
20	87.9015
30	90.0428
40	90.257
50	90.364
60	90.364
120	90.4711

Za algoritam LDA postotak točne klasifikacije je prikazan u tablici 4, što je slično točnosti PCA algoritma u slučaju detektiranih znakova, no nešto bolje za označene znakove.

Tablica 4: točnost klasifikacije znakova algoritmom LDA

	Točnost klasifikacije (%)
Označeni znakovi	92.7195
Detektirani znakovi	78.6938

Možemo primijetiti da za ovakav osnovni skup algoritam SVM u kombinaciji sa histogramom orijentacije gradijenta daje najbolje rezultate.

Pogledati ćemo još koliki je postotak točne klasifikacije za pojedine klase znakova. Za algoritam PCA uzeli smo dimenziju 40, pošto se ona pokazala najboljom za najviše klasa znakova.

Tablica 5: postotak točnosti klasifikacije detekcija za pojedine klase znakova

Klasa znaka	broj znakova skupa za treniranje	broj znakova skupa za testiranje	SVM (%)	PCA (%)	LDA (%)
A01	152	1	100	100	100
A03	42	128	92.1875	82.0313	88.2813
A04	24	165	59.3939	55.1515	63.6364
A05	32	156	90.3846	72.4359	80.7692
A07	4	12	33.3333	33.3333	41.6667
A08	194	38	100	97.3684	97.3684
A09	216	48	100	100	100
A10	203	24	87.5	91.6667	95.8333
A11	221	47	100	100	95.7447
A17	28	11	72.7273	90.9091	90.9091
A23	11	16	81.25	43.75	25
A25	18	6	50	50	16.6667
A33	31	206	84.466	82.0388	68.4466
A34	249	44	97.7273	88.6364	100
A44	214	32	100	96.875	100

U tablici 5 možemo vidjeti da je točnost klasifikacije za sve tri metode veća za klase koje imaju više znakova u skupu za treniranje. Klase koje imaju manje od 20 znakova, kao što su A07, A23 i A25 imaju uglavnom lošu točnost klasifikacije.

Ukoliko izbacimo iz skupa za učenje klase sa manje od 20 slika znakova točnost algoritma će se povećati, što možemo vidjeti u tablici 6.

Tablica 6: točnost klasifikacije detekcija za sve klase i samo za klase sa više od 20 znakova

	SVM(%)	PCA(%)	LDA(%)
Sve klase	84.4587	77.8373	78.6938
Bez malih klasa	86.5406	79.3333	81.1111

4.3. Učenje na stohastički pomaknutim slikama znakova

S ciljem povećanja točnosti algoritma prilikom klasifikacije detektiranih znakova skup za učenje smo promijenili na sljedeći način: svaki znak smo zamijenili sa četiri slike tog istog znaka, no pomaknutog na neki slučajni način.

Ideja je da se korištenjem normalne (Gaussove) razdiobe promijeni pozicija i dimenzija znaka. Normalna distribucija se označava kao (87), gdje je μ srednja vrijednost oko koje su raspršene vrijednosti, a σ^2 varijacija.

$$\mathcal{N}(\mu, \sigma^2) \tag{87}$$

Pošto imamo različite veličine znakova, neki su na slikama u daljini a neki blizu, dimenzije i poziciju znaka nije dobro mijenjati za sve znakove jednako, nego relativno u odnosu na njihovu veličinu. Poziciju i dimenzije znaka ćemo mijenjati relativno u odnosu na širinu znaka, što znači da će se uži znakovi (ujedno i manji) manje pomicati nego širi.

Pomicanje ćemo raditi na način da generiramo za svaki od četiri parametra, koordinate x i y opisanog pravokutnika te širinu i visinu pravokutnika, slučajan broj po zadanoj normalnoj razdiobi koji će se zatim množiti sa širinom slike te dodati zadanom parametru. Množenje sa širinom slike će osigurati da pomaci budu veći za veće slike i manji za manje.

Postupak ćemo isprobati na nekoliko različitih normalnih distribucija.

$$\mathcal{N}(0,0.02) \tag{88}$$

$$\mathcal{N}(0,0.03) \tag{89}$$

$$\mathcal{N}(0,0.04) \tag{90}$$

Za normalnu distribuciju sa parametrom σ^2 ćemo generirati najčešće brojeve između -2σ i 2σ. Tako za distribuciju sa parametrom $\sigma^2 = 0.03$ najčešće generiramo brojeve između -0.3464 i 0.3464. Uzmemo li npr. da je širina slike 40 piksela, svi parametri

se mogu promijeniti za do 14 piksela (40*0.3464), no zbog svojstva normalne razdiobe oni će češće biti oko nekoliko piksela.

Na slikama 19,20 i 21 se vide generirani znakovi korištenjem ove tri navedene normalne distribucije za veće i manje slike.

Slika 19: slike s generiranim pomacima normalnom razdiobom s parametrom 0.02

Slika 20: slike s generiranim pomacima normalnom razdiobom s parametrom 0.03

Slika 21: slike s generiranim pomacima normalnom razdiobom s parametrom 0.04

Slučajne brojeve po normalnoj distribuciju smo generirali korištenjem normalne distribucije implementirane u biblioteci Boost. Funkcija za inicijaliziranje normalne distribucije ima prototip:

```
normal_distribution(RealType mean = 0, RealType sd = 1);
```

gdje je `mean` srednja vrijednost μ, a `sd` varijacija σ^2. Osim toga potrebno je koristiti i:

```
variate_generator(Engine e, Distribution d);
```

koji služi za generiranje slučajnih brojeva korištenjem distribucije d, koja je u našem slučaju normalna distribucija.

Na taj način smo generirali tri skupa uzoraka, koji su sada četiri puta veći od originalnog skupa slika. Te slike su se dalje obrađivale na jednak način kao i u osnovnom postupku (odbacivanje linija, pretvorba u sivu sliku…)

U tablici 7 su prikazani postoci točnost algoritma SVM primijenjenom na tri generirana skupa i originalni skup. Možemo vidjeti da nam je ovakvo pomicanje pozicije i dimenzija slike poboljšalo uspješnost algoritma za detekcije znakova. S druge strane točnost za označene slike je manja. Pomak po normalnoj razdiobi sa parametrom $\sigma^2 = 0.02$ je ispao najoptimalniji za ovakav skup slika.

Tablica 7: točnost klasifikacije algoritmom SVM za različito pomaknute znakove skupa za treniranje

σ^2		Točnost klasifikacije (%)
0	Označeni znakovi	95.4936
	Detektirani znakovi	84.4587
0.02	Označeni znakovi	92.382
	Detektirani znakovi	86.6024
0.03	Označeni znakovi	92.382
	Detektirani znakovi	84.9946
0.04	Označeni znakovi	90.7725
	Detektirani znakovi	85.1018

U tablici 8 možemo vidjeti kako je pomicanje znakova utjecalo na točnost algoritma PCA. Jednako kao i kod SVMa pomicanje je poboljšalo točnost za detektirane znakove, ali je u ovom slučaju poboljšana točnost i za označene znakove.

Tablica 8: : točnost klasifikacije algoritmom PCA za različito pomaknute znakove skupa za treniranje

σ^2		Točnost klasifikacije (%)
0	Označeni znakovi	90.257
	Detektirani znakovi	77.8373
0.02	Označeni znakovi	93.2548
	Detektirani znakovi	85.439
0.03	Označeni znakovi	94.0043

	Detektirani znakovi	83.6188
0.04	Označeni znakovi	92.2912
	Detektirani znakovi	85.8672

Tablica 9 prikazuje točnost klasifikacije algoritmom LDA za pomaknute znakove. I u ovom slučaju su pomaci povećali točnost za detektirane znakove, no smanjili za označene znakove.

Tablica 9: : točnost klasifikacije algoritmom LDA za različito pomaknute znakove skupa za treniranje

σ^2		Točnost klasifikacije (%)
0	Označeni znakovi	92.7195
	Detektirani znakovi	78.6938
0.02	Označeni znakovi	92.0771
	Detektirani znakovi	83.2976
0.03	Označeni znakovi	89.0792
	Detektirani znakovi	81.1563
0.04	Označeni znakovi	82.7623
	Detektirani znakovi	75.4818

Iz rezultata se može vidjeti da je za sve tri metode najbolje uzeti pomak korištenjem normalne razdiobe sa $\sigma^2 = 0.02$.

Pošto je cilj pomicanja znakova bio povećati točnost postupka za detekcije znakova ovakva modifikacija skupa za treniranje je bila uspješna, iako je uglavnom smanjila točnost prilikom označenih znakova.

U sljedećem koraku probati ćemo umjesto četiri slike generirati osam slika. Prilikom učenja PCA i LDA algoritma se u ovom slučaju javljaju problemi sa nedostatkom memorije. Obje metode rade na način da se sve slike učitaju u matricu, nakon čega se računa kovarijacijska matrica, te iz nje svojstveni vektori i vrijednosti. Problem je u tome što se uslijed umnažanja slika osam puta kovarijacijska matrica povećala na veličinu od 1 GB, a matrice A i A^T na 0.5 GB, te program javlja da je ostao bez memorije. Kako bi riješili ovaj

problem uveli smo nekoliko promjena u program. Prva promjene je da umjesto da prvo računamo A^T i nakon toga množimo A i A^T, direktno računamo kovarijacijsku matricu iz matrice A, čime ćemo uštedjeti prostor koji je prije zauzimala matrica A^T. Druga je promjena to da ćemo podatke u programu prikazati umjesto u dvostrukoj preciznosti u jednostrukoj preciznosti.

U tablici 10 su prikazani rezultati testiranja kada su slike generirane normalnom razdiobom sa parametrom $\sigma^2 = 0.02$. Možemo vidjeti da generiranje osam slika nije donijelo neko značajnije povećanje u točnosti algoritma. Za metode PCA i LDA je točnost ostala otprilike jednaka, dok se za metodu SVM neznatno povećala.

Tablica 10: točnost klasifikacije pojedinih algoritama nakon generiranja osam slika za normalnu razdiobu $\sigma^2 = 0.02$

Metoda	Broj generiranih znakova	Točnost klasifikacije (%)
PCA	4	85.439
	8	84.1542
LDA	4	83.2976
	8	82.7623
SVM	4	86.6024
	8	88.9603

5. Zaključak

U ovom radu smo detaljnije proučili tri često korištene metode u klasifikaciji uzoraka: stroj s potpornim vektorom (SVM), analiza svojstvenih komponenti (PCA) i linearna diskriminantna analiza (LDA). Te tri metode imaju zajedničko da je za sve tri potrebno imati početni skup uzoraka koji se koristi za treniranje. Metoda SVM je u svome osnovnome obliku binarni klasifikator te svrstava uzorak u jednu od dvije moguće klase. S druge strane metode PCA i LDA u osnovi služe za smanjivanje dimenzije podataka uz najmanji mogući gubitak podataka. PCA to radi na način da minimizira razliku između početnih uzoraka i projiciranih. LDA s druge strane teži tome da klase projiciranih uzoraka budu što više razdvojive.

Pokazali smo da točnost navedenih metoda ovisi o količini slika u skupu za učenje, odnosno da znakovi iz onih klasa koje imaju više slika u skupu za učenje će biti točno klasificirani češće od znakova koji pripadaju klasama sa malo slika u skupu za učenje.

Za metodu PCA smo pokazali da točnost raspoznavanja ovisi o odabranom broju svojstvenih vektora, manja je ako se uzme premalo ili previše svojstvenih vektora. U slučaju premalo svojstvenih vektora nemamo dovoljno informacija za točnu klasifikaciju, a u slučaju previše vektora imamo previše nebitnih informacija koje unose grešku.

Pomicanjem slika u skupu za testiranje smo povećali učinkovitost algoritama za detekcije slika. Pomicanje slika je najviše utjecalo na metodu PCA gdje se povećala točnost za oko 7%, dok je kod LDA i SVM riječ o oko 5, odnosno 2% poboljšanja.

U ovako izvedenim testovima najboljom se pokazala metoda SVM, no uspješnost metoda ovisi o raznim faktorima. Odabir vektora značajki slike ima važan utjecaj na točnost algoritma, isto kao i dobra priprema slika. Pokazalo se da je korištenje histograma orijentacije gradijenta uspješna metoda računanja vektora značajki, te je to znatno povećalo učinkovitost klasifikacije metodom SVM.

6. Literatura

[1] Botiči I., Kovaček I., Kusalić I., Sustav za detekciju i raspoznavanje prometnih znakova, Fakultet elektrotehnike i računarstva, 2010

[2] Igor Bonači, Ivan Kusalić, Ivan Kovaček, Zoran Kalafatić, Siniša Šegvić., Addressing false alarms and localization inaccuracy in traffic sign detection and recognition. Computer Vision Winter Workshop, Mitterberg, Austria, February 2-4, 2011.

[3] Sambol A., Prepoznavanje objekata klasifikacijom histograma orijentacije gradijenta strojem s potpornim vektorima, Završni rad, Fakultet elektrotehnike i računarstva, 2010

[4] Šegvić S., Brkić K., Kalafatić Z., Stanisavljević V., Ševrović M., Budimir D., Dadić, I., A computer vision assisted geoinformation inventory for traffic infrastructure

[5] Sučić I., Fisherova linearna diskriminantna analiza, Seminar, Fakultet elektrotehnike i računarstva, 2010

[6] Sučić I., Primjena metode PCA nad skupom slika znakova, Završni rad, Fakultet elektrotehnike i računarstva ,2009

[7] „Support vector machine“, Wikipedia,the free encyclopedia, 6.12.2010, http://en.wikipedia.org/wiki/Support_vector_machine

[8] „Linear discriminant analysis“, Wikipedia, the free encyclopedia, 22.11.2010, http://en.wikipedia.org/wiki/Linear_discriminant_analysis

[9] „Principal component analysis“, Wikipedia, the free encyclopedia, 6.12.2010, http://en.wikipedia.org/wiki/Principal_component_analysis

[10] „Eigenvalues and eigenvectors“, Wikipedia, the free encyclopedia, 16.12.2010, http://en.wikipedia.org/wiki/Eigenvalue,_eigenvector_and_eigenspace

[11] „Jacobi eigenvalue algorithm“, Wikipedia, the free encyclopedia, 29.10.2010, http://en.wikipedia.org/wiki/Jacobi_eigenvalue_algorithm

[12] „Cholesky decomposition“, Wikipedia, the free encyclopedia, 25.11.2010, http://en.wikipedia.org/wiki/Cholesky_decomposition

[13] Kress, R., Numerical Analysis, Springer-Verlag, New York ,1998,str 126-133

[14] Open Source Computer Vision Library, Reference Manual, http://hawaiilibrary.net/eBooks/Give-Away/Technical_eBooks/OpenCVReferenceManual.pdf

[15] LIBSVM: a Library for Support Vector Machines, Chin-Chung Chang, Chih-Jen Lin, 2010

[16] ImageMagick Users Guide, E. I. du Pont de Nemours and Company, 1999, http://www.imagefolio.com/ImageMagick/ImageMagick.pdf

[17] Chang, C., Lin, C., LIBSVM: a library for support vector machines, 2001, http://www.csie.ntu.edu.tw/~cjlin/libsvm

[18] Chapelle, O., Haffner, P., Vapnik, V., SVMs for histogram-based image classification, IEEE Transactions on Neural Networks, vol 9, 1999

[19] Siegmann, P., Lafuente-Arroyo S., Maldonado-Bascon, S., Gil-Jimenez, P., Gomez-Moreno, H., Automatic evaluation of traffic sign visibility using SVM recognition methods, Proceedings oft he 5th WSEAS Int.Conf. on Signal Processing, Computational Geometry & Artificial Vision, Malta, September 15-17, 2005, str. 170-175, http://www.wseas.us/e-library/conferences/2005malta/papers/499-298.pdf

[20] Burges, C.J.C, A Tutorial on Support Vector Machines for Pattern Recognition, Dana Mining and Knowledge Discovery, vol 2., 1998, str 121-164

[21] M. Turk, A. Pentland, Eigenfaces for Recognition, Journal of Cognitive Neurosicence, Vol. 3, No. 1, 1991, str. 71-86

[22] K. Etemad, R. Chellappa, Discriminant Analysis for Recognition of Human Face Images, Journal of the Optical Society of America A, Vol. 14, No. 8, August 1997, str. 1724-1733

Naslov, sažetak i ključne riječi

Naslov

Eksperimentalna evaluacija metoda za prepoznavanje prometnih znakova

Sažetak

U ovom radu bavimo se primjenom triju često korištenih metoda za klasifikaciju uzoraka na prepoznavanje trokutastih znakova. Metode koje koristimo su: stroj s potpornim vektorima, analiza svojstvenih komponenti i linearna diskriminantna analiza. Opisani su načini rada tih metoda i programske implementacije koje smo koristili, zajedno sa pomoćnim metodama. Posebno smo ispitali mogućnost poboljšanja performansa modeliranjem lokalizacijske pogreške detekcijskih odaziva. Prikazali smo na nizu testova točnost pojedinih metoda i načine poboljšanja točnosti.

Ključne riječi

Prepoznavanje prometnih znakova, stroj s potpornim vektorima, analiza svojstvenih komponenti, linearna diskriminantna analiza

Title, abstract and keywords

Title

Experimental evaluation of different methods for traffic sign recognition

Abstract

In this paper we are using tree commonly used methods for sample classification to identify triangular signs. Methods are: support vector machine, principal component analysis and linear discriminant analysis. Algorithms of these methods are being described together with software implementation and additional methods we used. We present a series of tests to measure the accuracy of different methods and ways of improving accuracy.

Keywords

Traffic sign recognition, support vector machine, principal component analysis, linear discriminant analysis