SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1932

Strojno očitanje registarskih pločica na mobilnoj platformi

Srđan Rašić

Zagreb, srpanj 2011.

Zahvaljujem se mentoru doc. dr. sc. Siniši Šegviću na mentorstvu i stručnim savjetima kojima je pridonio ostvarenju ovog rada.

Sadržaj

1. Uvod

Mobilni računalni sustavi iz godine u godinu postaju sve napredniji, brži i dostupniji korisnicima. Zahtjevne poslove koje su donedavno mogla obavljati samo *glomazna* računala, danas vrlo uspješno obavljaju procesori veličine manje kovanice. Optički senzori visokih rezolucija postali su simbionti brzim niskopotrošnim procesorima. Takvi sustavi, upakirani u kućište manje od dlana ljudske ruke, stvaraju širok horizont mogućnosti primjene u polju računalnog vida.

Šire područje ovog rada je strojno očitanje automobilskih registarskih pločica sa slika. Strojno očitanje se sastoji od tri faze. Prva faza je dohvaćanje te obrada slike pločice da bi se dobio odgovarajući format koji zahtijevaju iduće faze. Zatim je potrebno segmentirati sliku, odnosno *izvući* sličice znakova (slova i brojeva) iz slike pločice. Posljednja faza predstavlja proces klasifikacije svih dobivenih znakova iz prethodne faze te prikaz rezultata korisniku.

Programsko ostvarenje rada izvedeno je na mobilnoj platformi. Konkretna platforma na kojoj je ostvarenje razvijano i testirano je mobilni uređaj *Apple iPhone 4* s operacijskim sustavom *iOS 4*. Uređaj je pogodan za implementaciju ovog rada jer pruža odlične performanse procesora te optički senzor visoke rezolucije. Dizajniran je na sklopovlju koje predstavlja dobar uzorak današnjih pametnih telefona te snage mobilnih platformi. Stoga bi se implementacija rada, uz male promjene, mogla prebaciti na *Android* ili *Windows Phone 7* uređaje bez značajne razlike u uspješnosti očitanja pločica.

Uspješnost očitanja ovisi o raznim čimbenicima kao što su kvaliteta ulaznih slika, varijabilna količina osvjetljenja na slici, nečistoća pločica, performanse mobilne platforme i slično. U ovom radu navedeni su najčešći problemi do kojih dolazi prilikom očitanja te njihova moguća rješenja.

2. Obrada ulaznih slika

Da bi se slika registarske pločice mogla očitati, potrebno ju je obraditi i pronaći sve znakove koji se onda prosljeđuju u idući fazu. Obrada slika sastoji se od nekoliko koraka. Prvi korak je binarizacija slike. Slijedi segmentacija te izrezivanje segmentiranih dijelova. Na ulazu se nalazi slika sa pločicom. Rezultat obrade su binarizirane sličice znakova.

Slika 2.1: Faza obrade slika

2.1. Zapis slike u memoriji računala

Da bi se provela segmentacija, potrebno je binarizirati slike. Rasterske slike u memoriji računala zapisane su kao niz slikovnih elemenata (piksela) određene dimenzionalnosti. Dimenzionalnost, koja je rezultat računalnog zapisa boje, omogućava apstraktnu zamisao slike kao skupa *stopljenih* kanala. Kanal predstavlja zapis cijele slike s pripadnim vrijednostima nijansi boje koju kanal predstavlja. Uobičajena slika u boji, fotografija, sastoji se od tri kanala; crvenog, zelenog i plavog (engl. *Red, Green, Blue; RGB*). Stapanjem tih triju kanala, na zaslonu se

dobiva slika u stvarnim bojama (slika 2.2). Slikovni elementi su određeni i dubinom boje (engl. *color depth*), odnosno brojem bitova koje imaju na raspolaganju za prikaz boje. Digitalne kamere na mobilnim platformama uglavnom stvaraju 24-bitne slike. To znači da je svaki slikovni element određen sa količinom informacije od 24 bita. Proporcionalno raspoređeno po kanalima dobiva se 8 bitova po kanalu. 8-bitni binarni zapis jednoznačno određuje $2^8 = 256$ brojeva što znači da svaki kanal po slikovnom elementu razlikuje 256 nijansi određene boje. Kombiniranjem triju takvih kanala ostvaruje se zapis koji razlikuje $256^3 = 16\ 777\ 216$ boja.

Slika 2.2: Stapanje kanala

Toliko boja (odnosno nijansi svih boja) potrebno je da bi čovjek imao realan doživljaj prilikom promatranja slike, no taj broj je nepotrebno, pa čak i destruktivno, prevelik za vrstu posla kojom se bavi ovaj rad – raspoznavanje znakova (slova i brojki). Naime, da bi prepoznao znak, čovjeku ili računalu, dovoljne su samo dvije različite boje. To mogu biti bilo koje boje, ali radi jednostavnosti uzimaju se crna i bijela.

2.2. Binarizacija

Proces smanjenja broja boja na dvije boje (a time i na jedan kanal) naziva se binarizacija. Na ovaj proces se može gledati kao na prvi korak segmentacije jer se važniji slikovni elementi koji tvore objekt koristen u kasnijim fazama razdvajaju od pozadine (Wikipedia, 2011c). Ideja je vrlo jednostavna – tamnije dijelove slike obojiti jednom bojom (obično crnom), a svjetlije dijelove drugom bojom (bijelom). Pritom je potrebno obojiti cijelu površnu slike. Postavlja se pitanje kako odrediti koji dijelovi su tamniji (objekt), a koji svjetliji te u odnosu na što su oni tamniji odnosno svjetliji (pozadina). Uvodi se pojam *vrijednost praga*. Vrijednost praga predstavlja referentnu vrijednost intenziteta boje. Ukupni intenzitet elementa računa se kao suma intenziteta svih triju boja skaliranih određenim

udjelom. Standardni udjeli (k_r, k_g i k_b) su 30% crvene boje, 59% zelene boje i 11% plave boje no za potrebe ovog projekta drugačiji udjeli daju bolje rezultate (Wikipedia, 2011b). Obzirom da je grb na pločici uglavnom crvene boje, uzima se 100% intenziteta crvene boje, a 0% intenziteta ostalih boja. Dakle, $k_r = 1$ i $k_g = k_b = 0$. Time se postiže uklanjanje većeg djela grba što je poželjno jer je grb neželjeni objekt kao što je objašnjeno kasnije. Svaki element tamniji od praga svrstava se pod objekt, dok se svaki svjetliji smatra pozadinskim.

Algoritam 1 Binarizacija slike

1. Postavi vrijednost praga $threshold = \text{CONST}$.
2. Postavi udjele boja k_r, k_g i k_b na definirane vrijednosti.
3. Za svaki element slike napravi sljedeće:
   (a) Neka je $grey = red_{i,j} * k_r + green_{i,j} * k_g + blue_{i,j} * k_b$
   (b) Ako je $grey \leq threshold$
      – onda $pixel_{i,j}$ stavi u skup *objekt*
      – inače $pixel_{i,j}$ stavi u skup *pozadina*
4. Stvori binariziranu sliku bojajući elemente s indeksima jednakim indeksima elemenata u skupu *objekt* crnom bojom, a elemente s indeksima jednakim indeksima elemenata u skupu *pozadina* bijelom bojom.

Algoritam 1 opisuje proces binarizacije slike u boji. Svaki element se postavi na jednu od dvije boje ovisno o vrijednosti praga. Odmah se može uočiti nedostatak algoritma – ograničenost predefiniranom nepromjenjivom vrijednosti praga. Uspješnost binarizacije ovisi o vrijednosti praga i osvjetljenju slike. Postoji nekoliko načina za određivanje vrijednosti praga, no u ovom slučaju sasvim je dovoljna jednostavna iterativna metoda čiji opis je prikazan u algoritmu 2. Ideja je da se iterativno traži vrijednost praga koja konvergira ka nekoj konačnoj vrijednosti. Ta vrijednost je rezultat algoritma.

Rezultat binarizacije je slika reducirana na dvije boje koje jednoznačno razdvajaju objekt (slova, brojke i dodatne oblike poput grba ili okvira pločice) od pozadine (šuma). Takav prikaz pogodan je za iduću fazu – segmentaciju.

Algoritam 2 Određivanje vrijednosti praga iterativnom metodom

1. Postavi vrijednost praga T na neki određeni ili nasumični broj.
2. Binariziraj sliku po algoritmu 1 uz vrijednost praga T. Binarizacijom se stvaraju dva skupa, *objekt* i *pozadina*.
3. Izračunaj srednje vrijednosti skupova:
   (a) Neka je m_1 = aritmetička sredina skupa *objekt*
   (b) Neka je m_2 = aritmetička sredina skupa *pozadina*
4. Izračunaj novu vrijednost praga na osnovu m_1 i m_2:
   (a) Neka je $T' = (m_1 + m_2)/2$
5. Vrati se na korak 2 i ponovno binariziraj sliku koristeći T'. Ponavljaj dok nova vrijednost praga ne bude jednaka prethodnoj.

2.3. Segmentacija

Slika 2.3: Pravokutni okvir objekta (narančasto)

Pojam segmentacija označava proces izdvajanja pojedinačnih objekata iz ulazne slike. Objekt u ovoj fazi treba razlikovati od objekta u procesu binarizacije. U slučaju slike registarske pločice objekti su slova, brojke i ostali oblici koji se zanemaruju. Objekt je na slici određen minimalnim pravokutnim okvirom koji ga obuhvaća (slika 2.3). Nakon što se odredi okvir, objekt se iz slike pločice *izvlači* tako da se od dijela slike određenog okvirom stvori nova sličica. Postoji mnogo postupaka segmentacije. Uz pretpostavku da je slika koja se dovodi na ulaz ove faze čista (da nema šuma) te da se slova i brojke dobro razlikuju, moguće je implementirati vrlo jednostavan algoritam koji segmentira sliku. Ova pretpostavka ne predstavlja jako ograničenje jer binarizacija slike pločice ostvarena algoritmom u prethodnoj fazi općenito daje dobar rezultat – čistu sliku bez šuma s jasno izraženim znakovima.

2.3.1. Fizikalni opis registarske pločice

Da bi bilo moguće definirati korisnu informaciju koju sadržava registarska pločica prvo ju je potrebno odrediti. U ovom projektu, radi jednostavnosti, odabrana je samo jedna vrsta registarskih pločica; standardne registarske pločice osobnih automobila. Takve pločice određene su dimenzijama 520x110 mm. Pločica je bijele boje, a na nju je otisnut tekst crne boje i grb Republike Hrvatske. Tekst se nalazi na sredini pločice i podijeljen je u tri dijela. Prvi dio sastoji se od dva slova nakon kojih je otisnut grb. Drugi dio sastoji se od tri ili četiri brojke. Treći dio sadržava jedno ili dva slova. Drugi i treći dio odvojeni su crticom (povlakom). Ispred i iza teksta nalazi se prazan (bijeli) prostor. Širina tog prostora nikad nije manja od 5% širine pločice.

2.3.2. Algoritam

Algoritam obrađuje sliku registarske pločice. Uvjet za ispravan rad algoritma je slika na ulazu koja sadrži isključivo pločicu – ništa više od pločice (npr. stražnji dio automobila, svjetla i slično) i ništa manje od pločice (slika gdje je bilo koji dio pločice, uključujući i bijeli prostor, odrezan). Dozvoljeno je odstupanje koje se odnosi na okvir (držač) pločice.

Pretpostavimo da se prvo slovo teksta pločice ne nalazi lijevo od točke koja određuje lijevih 5% širine slike. Dozvolimo da se skroz lijevo ili skroz desno na slici nalaze neželjeni objekti poput komada okvira ili držača registarske pločice i pretpostavimo da njihova širina ne prelazi 5% širine slike. Pretpostavimo nadalje da se tekst pločice, promatrajući samo vertikalnu os, nalazi na sredini slike. Ovim pretpostavkama definirali smo interval horizontalne dimenzije područja slike unutar kojeg se mora nalaziti tekst pločice te njegovo vertikalno središte. Korisno je zamisliti liniju koja prolazi vertikalnim središtem slike te počinje u točki koja određuje lijevih 5% širine slike, a završava u točki koja određuje desnih 5% širine slike. Neka se ta linija zove *linija skeniranja*.

Ideja algoritma je sljedeća. Neka je skup *segmenti* prazan skup. Pomicati se po liniji skeniranja, element po element, od početne točke udesno sve dok se ne naiđe na crni slikovni element odnosno objekt. Odrediti granični okvir objekta. Okvir dodati u skup *segmenti*. Nastaviti se kretati po liniji skeniranja od točke koju definira desna granica prethodno određenog okvira sve dok se ne dođe do idućeg objekta kojeg je potrebno obraditi kao i prethodni. Proces završava kada se dođe do kraja linije skeniranja. Rezultat ovog postupka je skup okvira koji

određuju položaje objekata na slici registarske pločice.

Preostaje još definirati postupak određivanja graničnog okvira objekta. Vrlo efikasno rješenje temelji se na kruženju oko objekta. Nakon što se pronađe jedna rubna točka objekta, potrebno se je gibati, element po element, rubom objekta u jednom smjeru (npr. u smjeru kazaljke na satu) sve dok se ponovno ne dođe do točke iz koje se krenulo (slika 2.4). U svakoj iteraciji računa se minimalna i maksimalna horizontalna i vertikalna granica – te četiri granice određuju granični pravokutni okvir objekta.

Slika 2.4: Određivanje okvira kruženjem oko objekta

Ovako definiran algoritam *traženja* objekata unutar slike suočava se s problemom *neželjenih objekata* koji su vrlo česti. Jedan od tih objekata je grb, odnosno njegovi dijelovi. Naime, binarizacija obično uklanja većinu površine grba obzirom da je on puno svjetliji od slova i brojki te zbog korištenih nestandardnih omjera kanala prilikom binarizacije, no neke njegove dijelove ipak ostavi. Ti dijelovi su obično pojedinačni kvadratići, skupine kvadratića ili kruna grba. Često se kao objekt prepozna cijeli rub (držač) pločice. Prljava ili oštećena pločica također rezultira neželjenim objektima. Srećom, svi navedeni objekti imaju jednu značajku kojom se znatno razlikuju od slova i brojki. To je njihova visina. Visine preostalih dijelova grba nakon binarizacije su redovito za više od dvostruko manje od slova. Nečistoća i oštećenja također. Okvir (držač) pločice je od slova viši za najmanje 50%. Postavljanjem ograničenja visine objekta iz skupa *segmenti* izbacuje se većina neželjenih objekata. Ako se dogodi da koji neželjeni objekt ipak izbjegne eliminaciju, on će kasnije rezultirati pogrešnim očitanjem pločice te će se takvo očitanje smatrati neuspjehom. Zbog smanjenja kompleksnosti projekta,

rješavanje takvih problema nije razmatrano.

Rezultat ovog koraka su sličice objekata (slova i brojki) koje su iz slike pločice izvučene opisanim postupcima. Time je dovršena faza obrade slike registarske pločice. Pronađeni objekti se zatim prosljeđuju idućoj fazi – fazi klasifikacije, odnosno prepoznavanje znakova.

3. Raspoznavanje znakova s registarske pločice

Prepoznavanje znakova (slova i brojki) problem je prepoznavanja uzoraka. Na ulazu se nalazi slika slova ili brojke iz koje treba *iščitati* informaciju, odnosno slovo ili brojku koja je na njoj prikazana. Jednostavna, ali dosta moćna, metoda klasifikacije je algoritam *K-najbližih susjeda*. Algoritam se zasniva na traženju najbližih primjera iz skupa klasificiranih primjera za učenje. Primjeri mogu biti bilo koji uzorci podataka. U ovom slučaju su to slike, no kako one sadrže nepotrebno puno podataka o informaciji koju prikazuju, potrebno je izlučiti najvažnije značajke tih podataka. Postupak za to implementiran u ovom radu naziva se *Analiza svojstvenih komponenata* (engl. *Principal component analysis, PCA*).

3.1. Matematičke osnove

Prije formalne definicije *PCA* postupka, potrebo je objasniti matematičke pojmove koje postupak koristi.

3.1.1. Važniji statistički pojmovi

Statistika objedinjuje ideje i postupke koji se mogu primijeniti nad skupom podataka. Cilj joj je analizirati odnose između individualnih elemenata skupa podataka.

Aritmetička sredina je jedna od središnjih vrijednosti koje se koriste u statistici. Računa se za neki skup brojeva kao kvocijent zbroja elemenata i broja tih elemenata.

$$\overline{x} = \frac{\sum_{i=1}^{n} x_i}{n} \tag{3.1}$$

Nažalost, aritmetička sredina ne govori puno podacima. Dva skupa mogu imati jednaku aritmetičku sredinu, ali su očito različiti:

$$(0, 8, 12, 20) \text{ i } (8, 9, 11, 12)$$

Postavlja se pitanje što je različito u ovim skupovima. Radi se o *raspršenosti* podataka. Podaci mogu biti *blizu* aritmetičkoj sredini, ali mogu biti i jako *udaljeni* od nje. Mjera raspršenosti naziva se *standardna devijacija*. Računa se kao srednja udaljenost elementa od aritmetičke sredine skupa po formuli 3.2.

$$s = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \overline{x})^2}{n}} \tag{3.2}$$

Varijanca je još jedna mjera raspršenosti podataka. Gotovo je identična standardnoj devijaciji kao što se vidi u formuli 3.3.

$$s^2 = \frac{\sum_{i=1}^{n}(x_i - \overline{x})^2}{n} \tag{3.3}$$

Standardna devijacija i varijanca su mjere jednodimenzionalnih skupova podataka kao što su ocjene iz ispita, dob zaposlenika itd. Međutim, skupovi podataka često imaju više od jedne dimenzije. Cilj statističke analize tih skupova je obično pokušaj utvrđivanja odnosa između dimenzija. Neka je dan skup podataka koji sadrži vremena koliko je koji student učio i ocjene koje su dobili iz ispita. Možda bi bilo poželjeno utvrditi ima li vrijeme učenja utjecaj na njegovu ocjenu. Mjera koja se koristi za utvrđivanje tog odnosa naziva se *kovarijanca*. Računa se uvijek između dvije dimenzije. Ako se računa između jedne dimenzije i same sebe, dobiva se varijanca. Formula za varijancu može se napisati i ovako:

$$var(x) = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(x_i - \overline{x})}{n} \tag{3.4}$$

Obzirom na opisanu definiciju kovarijance, iz te formule se jednostavno dobiva formula za računanje kovarijance:

$$cov(x, y) = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{n} \tag{3.5}$$

Pokažimo na primjeru što predstavlja kovarijacija. Recimo da imamo dvodimenzionalni skup podataka. Jedna dimenzija, S, neka sadrži broj sati učenja, a druga, B, broj bodova koje je student dobio. Tablica 3.1 prikazuje izmišljene podatke i izračun kovarijacije između broja sati i broj bodova. Sama vrijednost kovarijacije nije toliko važna koliko je njezin predznak. Pozitivna vrijednost *kaže*

da se vrijednosti iz obje dimenzije povećavaju – kako se povećava broj sati učenja, povećava se i broj ostvarenih bodova. Ako je vrijednost negativna, znači da se vrijednost iz jedne dimenzije smanjuje, a iz druge povećava.

S	B	$(S_i - \overline{S})$	$(B_i - \overline{B})$	$(S_i - \overline{S})(B_i - \overline{B})$
9	39	-4.92	-23.42	115.23
15	56	1.08	-6.42	-6.93
25	93	11.08	30.58	338.83
14	61	0.08	-1.42	-0.11
10	50	-3.92	-12.42	48.69
18	75	4.08	12.58	51.33
0	32	-13.92	-30.42	423.45
16	85	2.08	22.58	46.97
			Ukupno	1017.46
			Prosjek	127.18

Tablica 3.1: Primjer skupa podataka

Kako se kovarijanca računa između dvije dimenzije potrebno je osmisliti zapis koji će sadržavati sve moguće kovarijance između svih dimenzija. Jedan od načina da se to ostvari je staviti sve različite kovarijance u matricu. Takva matrica naziva se *kovarijacijska matrica*. Primjer kovarijacijske matrice za dvodimenzionalni skup prikazan je ispod.

$$C = \begin{pmatrix} cov(x,x) & cov(x,y) \\ cov(y,x) & cov(y,y) \end{pmatrix}$$

Općenita formula kovarijacijske matrice *n-dimenzionalnog* skupa dana je izrazom:

$$C = \begin{pmatrix} cov(x_1,x_1) & cov(x_1,x_2) & \dots & cov(x_1,x_n) \\ cov(x_2,x_1) & cov(x_2,x_2) & \dots & cov(x_2,x_n) \\ \dots & \dots & \dots & \dots \\ cov(x_n,x_1) & cov(x_n,x_2) & \dots & cov(x_n,x_n) \end{pmatrix} \tag{3.6}$$

3.1.2. Svojstveni vektori i svojstvene vrijednosti

Dvije matrice, ukoliko su odgovarajućih dimenzija, moguće je pomnožiti. Specifičan slučaj tog množenja definira svojstvenu vrijednost i svojstvene vek-

tore. Neka je A proizvoljna kvadratna matrica dimenzija $n \times n$, vektor $\vec{v}$ n-dimenzionalan vektor i λ proizvoljan skalar. Skalar λ naziva se *svojstvena vrijednost* matrice A, a vektor $\vec{v}$ naziva se *svojstveni vektor* matrice A ako vrijedi izraz:

$$A\vec{v} = \lambda\vec{v} \tag{3.7}$$

3.2. Analiza svojstvenih komponenata

Analiza svojstvenih komponenata (engl. Principal Components Analysis) je postupak identificiranja uzoraka u podacima ističući sličnosti i razlike u njima (Smith, 2002). To je također način sažimanja podataka tako da im se smanji dimenzionalnost bez većih gubitaka informacije sadržane u njima. Postupak se može podijeliti u nekoliko koraka opisanih u nastavku.

3.2.1. Postupak

1. Priprema podataka. Da bi se proveo *PCA* postupak potreban je neki skup podataka. Taj skup podataka treba biti organiziran u više dimenzija jednakih veličina. Od skupa stvara se matrica nad kojom će se provoditi postupak tako da se svaka dimenzija skupa zapiše u jedan redak matrice. Neka je ta matrica označena simbolom A.

2. Računanje i oduzimanje srednje vrijednosti. Da bi postupak analize svojstvenih komponenata uspješno radio, potrebno je pronaći srednju vrijednost svake dimenzije matrice A. Srednja vrijednost se računa kao aritmetička sredina. Zatim se svakom elementu dimenzije za koju je izračunata srednja vrijednost oduzme ta vrijednost. Rezultat je skup podataka čija je aritmetička sredina 0. Vektor koji sadržava srednje vrijednosti svake dimenzije zove se *srednja slika.*

3. Računanje kovarijacijske matrice. Matrica kovarijacije C vrlo se lako može dobiti umnoškom matrice A i transponirane matrice A' te dijeljenjem svakog elementa matrice sa brojem dimenzija n.

(i, j)-ti element umnoška matrice A dimenzija $n \times m$ i njene transponirane matrice definiran je izrazom

$$X_{i,j} = \sum_{k=1}^{m} A_{i,k} A_{j,k}.$$

Umnožak X matrica je dimenzija $m \times m$. Kovarijanca između i-te i j-te dimenzije (retka) matrice A računa se prema izrazu 3.5.

$$cov(i,j) = \frac{\sum_{k=1}^{m}(A_{i,k} - \overline{A_i})(A_{j,k} - \overline{A_j})}{n}.$$

No kako je matrici A već oduzeta srednja vrijednost (slika), taj izraz prelazi u

$$cov(i,j) = \frac{\sum_{k=1}^{m} A_{i,k}A_{j,k}}{n},$$

što se može zapisati kao

$$cov(i,j) = \frac{1}{n}X_{i,j}$$

što prema izrazu 3.6 daje konačnu formulu za računanje kovarijacijske matrice matrice A.

$$C = \frac{1}{n}(AA') \tag{3.8}$$

4. Određivanje svojstvenih vrijednosti i vektora. Izračunatoj kovarijacijskoj matrici C potrebno je odrediti svojstvene vrijednosti i svojstvene vektore. Obzirom da je matrica C kvadratna matrica, to je moguće. Radi kompleksnosti postupka računanja svojstvenih vrijednosti i svojstvenih vektora taj postupak ovdje nije objašnjen niti je implementiran u ovom radu već je korištena funkcija iz gotove *alglib* biblioteke. Svojstvene vrijednosti zapisuju se u matricu S dimenzija $1 \times n$, a svojstveni vektori u matricu V dimenzija $n \times n$ gdje je n dimenzionalnost matrice C odnosno broj svojstvenih vrijednosti ili vektora. Svojestvene vrijednosti uparene su sa svojstvenim vektorima tako da i-ti svojstvena vrijednost odgovara svojstvenom vektoru u i-tom stupcu matrice V.

5. Stvaranje vektora značajki. Vektor kojemu odgovara najveća svojstvena vrijednost naziva se *svojstvena komponenta* skupa. Ovaj korak podrazumijeva odabir p svojstvenih vektora koji odgovaraju najvećim svojstvenim vrijednostima. Time se gubi dio informacije, ali taj dio je puno manjeg značaja od onog koji se zadržava. Od tih vektora formira se *vektor značajki* što je samo zgodan naziv za matricu vektora.

$$W = \begin{pmatrix} sv_{1_1} & sv_{2_1} & \dots & sv_{p_1} \\ sv_{1_2} & sv_{2_2} & \dots & sv_{p_2} \\ \dots & \dots & \dots & \dots \\ sv_{1_n} & sv_{2_n} & \dots & sv_{p_n} \end{pmatrix}$$

6. Reduciranje dimenzionalnosti izvornih podataka. Posljednji korak *PCA* postupka je ujedno i najlakši. Vektor značajki izračunat u prethodnom koraku potrebno je transponirati te ga s lijeva pomnožiti sa izvornim podacima.

$$R = W^T A$$

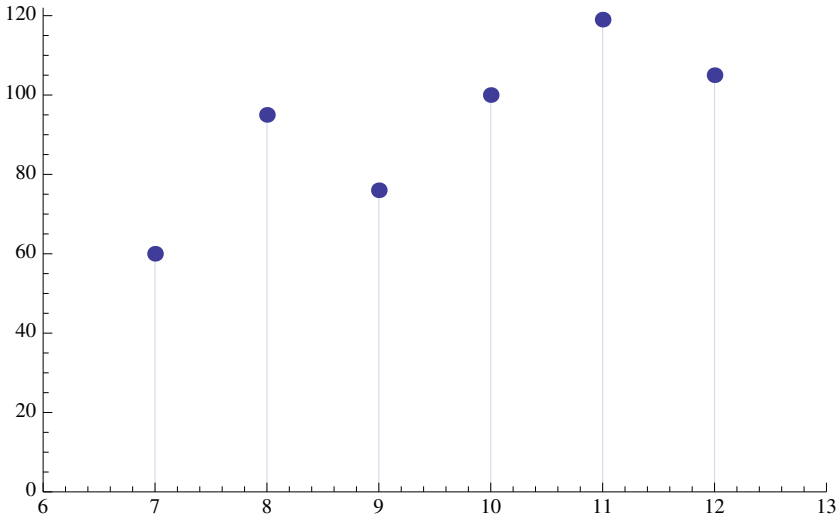
Umnožak predstavlja izvorne podatke prikazane u odnosu na odabrane vektore. Ti podaci su smanjene dimenzionalnosti. Cijena toga je gubitak informacija, no taj gubitak je zanemariv u odnosu na prednosti koje se pružaju korištenjem takvih podataka u *k-nn* postupku.

3.2.2. Postupak *PCA* na intuitivnom primjeru

Iako je ovaj postupak prilično lako implementirati, malo je teže shvatiti ga. Korisno je proučiti postupak na primjeru. Neka je skup podataka koji treba analizirati dvodimenzionalan. Prednost proučavanja takvog skupa je što se može prikazati grafički. Podaci su prikazani u tablici 3.2 te na grafu 3.1. Kao što se može uočiti, jedna dimenzija sadržava podatke malih vrijednosti s malim razlikama među podacima, a druga podatke većih vrijednosti s većim razlikama. Ovo je važno uočiti jer je za *PCA* postupak to ključna značajka.

X	10	12	7	8	9	11
Y	100	105	60	95	76	119

Tablica 3.2: Podaci za analizu



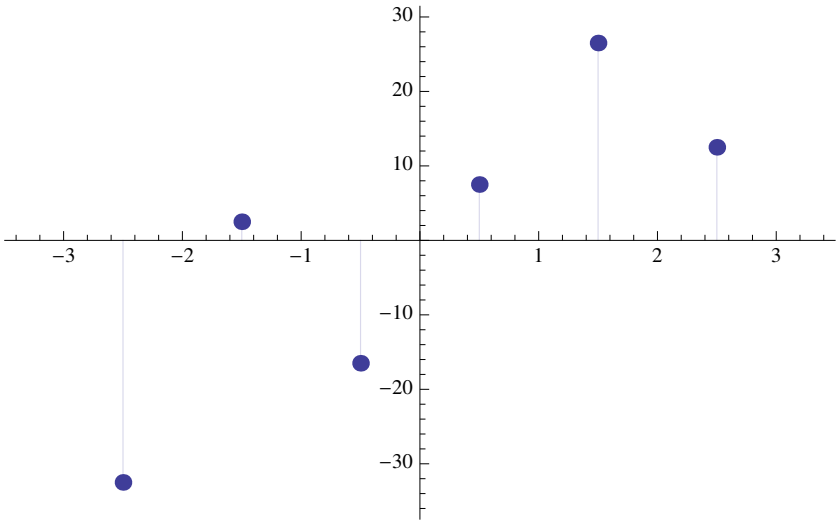
Slika 3.1: Podaci za analizu

Ulazne podatke potrebno je centrirati oko ishodišta. Zato se svakoj dimenziji oduzima njezina aritmetička sredina. Rezultat oduzimanja zapisuje se u matricu A tako da se svaka dimenzija zapiše u poseban redak.

$$\overline{A} = \begin{pmatrix} 9.5 \\ 92.5 \end{pmatrix}$$

$$A = \begin{pmatrix} 0.5 & 2.5 & -2.5 & -1.5 & -0.5 & 1.5 \\ 7.5 & 12.5 & -32.5 & 2.5 & -16.5 & 26.5 \end{pmatrix}$$

Centrirani podaci mogu se prikazati grafički kao što se vidi na slici ispod. Može se uočiti da su odnosi među podacima ostali nepromijenjeni.



Slika 3.2: Podaci centrirani oko ishodišta

Sljedeći korak podrazumijeva računanje kovarijacijske matrice matrice A. Kovarijacijska matrica računa se po formuli 3.8. Obzirom da su podaci dvodimenzionalni, kovarijacijska matrica će biti dimenzija 2×2.

$$C = \begin{pmatrix} 8.75 & 80.25 \\ 80.25 & 1124.75 \end{pmatrix}$$

Matrica je simetrična jer je kovarijanca između dimenzije X i dimenzije Y jednaka kovarijanci između dimenzije Y i dimenzije X. Takvo svojstvo proizlazi iz formule za kovarijancu (3.5). Nadalje, vrijednosti nedijagonalnih elemenata su pozitivne što znači da bi se podaci obiju dimenzija trebali zajedno kretati (povećavati ili smanjivati) što ovom primjeru vrijedi.

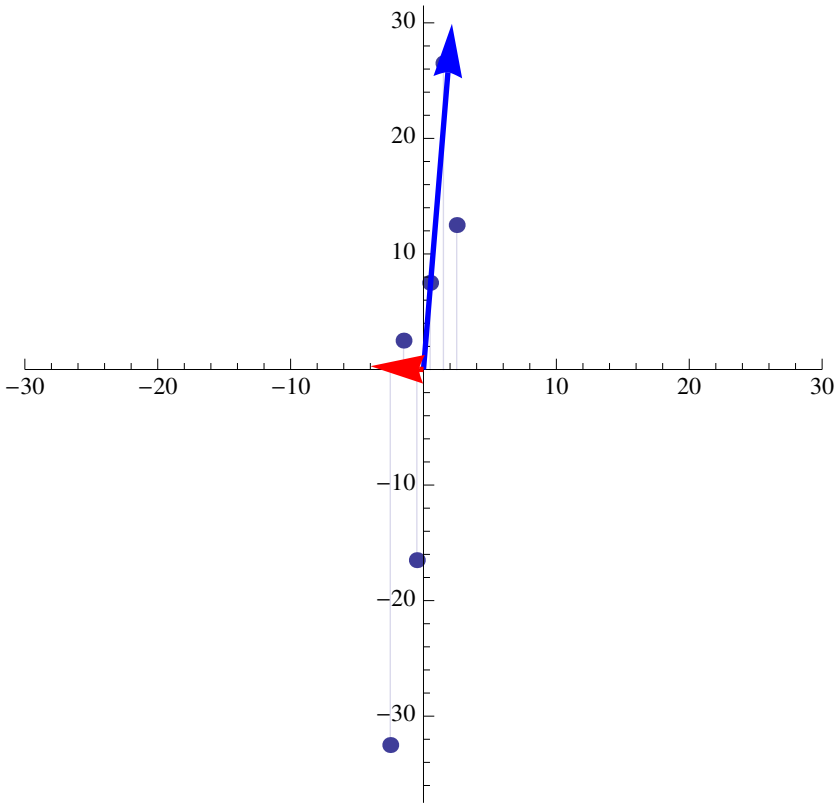
Postupkom analize svojstvenih komponenata iz podataka se izdvajaju najvažnije informacije o njime. Osnovni korak u tome je računanje svojstvenih vektora i svojstvenih vrijednosti kovarijacijske matrice. Izračunati vektori i vrijednosti matrice C prikazani su ispod. Svojstveni vektori zapisani su u stupcima te su normalizirani.

$$SvojstveneVrijednosti = \begin{pmatrix} 1130.49 & 3.00887 \end{pmatrix}$$

$$SvojstveniVektori = \begin{pmatrix} 0.0713582 & -0.997451 \\ 0.997451 & 0.0713582 \end{pmatrix}$$

Na slici 4.5 može se vidjeti da podaci imaju vrlo izražajan uzorak. Na slici su iscrtani i svojstveni vektori. Vidljivo je da su oni okomiti jedan na drugog, ali što je važnije, vidljivo je da pružaju informaciju o uzorku podataka. Jedan od vektora *prolazi* kroz *središte* podataka. Moglo bi se reći da stvara pravac koji najbolje opisuje podatke. Drugi, manje važan, vektor pokazuje da su elementi skupa koji prate pravac malo odmaknuti od njega.

Ovim korakom dobiveni su vektori (pravci) koji karakteriziraju skup podataka. Idući korak je transformirati podatke tako da budu izraženi pomoću tih vektora.



Slika 3.3: Svojstveni vektori

Svojstveni vektor s najvećom svojstvenom vrijednosti naziva se *svojstvena komponenta* skupa. U ovom primjeru to je vektor koji *prolazi* kroz središte podataka.

Vektor značajki stvara se tako da se odabere p svojstvenih vektora s najvećim svojstvenim vrijednostima. Premda ovaj primjer sadrži samo dva svojstvena vektora, bit će odabran samo onaj s najvećom svojstvenom vrijednosti.

$$W = \begin{pmatrix} 0.0713582 \\ 0.997451 \end{pmatrix}$$

Taj vektor služi za transformaciju izvornih podataka. Transformacija se obavlja jednostavnim množenjem transponiranog vektora značajki matricom sa izvornim podacima centriranim oko ishodišta.

$$\begin{aligned} R &= W^T A \\ &= \begin{pmatrix} 7.51 & 12.64 & -32.59 & 2.38 & -16.49 & 26.53 \end{pmatrix} \end{aligned}$$

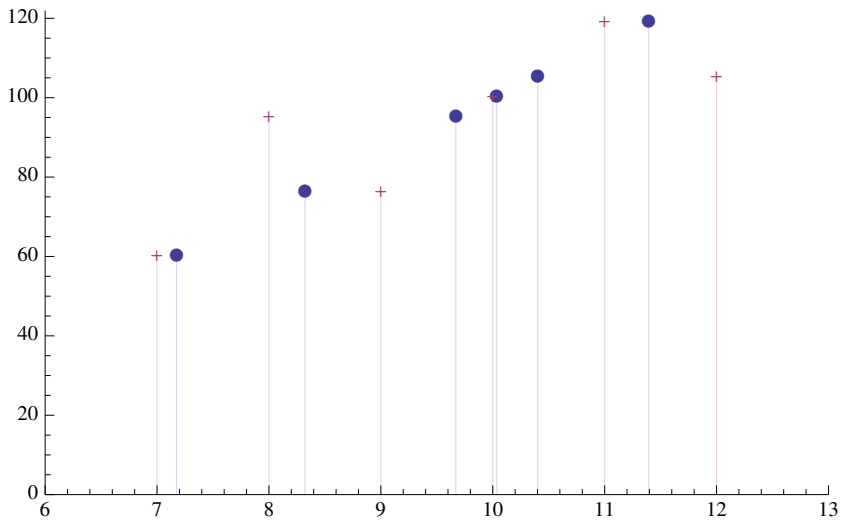
Dobiveni skup sadržava podatke izražene u odnosu na odabrane vektore u transformiranom prostoru. Odabrani vektori su oni koji najbolje opisuju podatke, odnosno oni koji opisuju najvažnije značajke podataka – kojima su pridružene najveće svojstvene vrijednosti. Obzirom da je odabran samo jedan vektor, izgubljen je dio informacije. To je informacija koju opisuje drugi vektor – može se zamisliti kao udaljenost svakog pojedinog elemenata skupa od linije koju određuje značajniji vektor. Kako bi se odredila količina izgubljene informacije, potrebno je dobivene podatke obrnutim postupkom transformirati u podatke slične izvornim. Jednostavnim izvodom dolazi se do formule za to.

$$ObnovljeniPodaci = WR + \overline{A}$$

Za korišteni primjer to će dati sljedeće podatke.

$$\begin{pmatrix} 10.0364 & 10.4024 & 7.17404 & 9.6703 & 8.32305 & 11.3938 \\ 99.9974 & 105.114 & 59.9875 & 94.8805 & 76.0484 & 118.972 \end{pmatrix}$$

Iscrtavanjem na grafu uočava se razlika u odnosu na izvorne podatke. Varijacija uz svojstvenu komponetu (vektor) je sačuvana, dok je varijacija uz drugi vektor izgubljena.



Slika 3.4: Obnovljeni izvorni podaci (•) i izvorni podaci (+)

3.2.3. *PCA* nad slikama registarskih pločica

Skup podataka nad kojim se u ovom radu provodi postupak analize svojstvenih komponenata je skup slika za učenje. Svaka od tih slika prikazuje jedan znak

(slovo ili brojku) i obrađena je postupkom opisanom u poglavlju 2. Slike su kvadratne te skalirane na jednake veličine.

Svaka slika se može zapisati u obliku *n-dimenzionalnog* vektora odnosno jednostupčane matrice od n redaka gdje je n umnožak širine i visine slike tako da se nanižu reci slikovnih elemenata jedan ispod drugog. Elementi matrice (vektora) su vrijednosti boje svakog slikovnog elementa. U slučaju binarizirane slike to su samo vrijednosti 0 (crna) i 255 (bijela). Od takvih vektora se stvara $m \times n$ matrica tako da se svaki od m vektora (slika) zapiše kao jedan stupac matrice. To je matrica A koja predstavlja skup podataka nad kojim se provodi analiza svojstvenih komponenata. Rezultat analize svojstvenih komponenata je matrica A, odnosno skup slika, smanjene dimenzionalnosti. Ovim postupkom dimenzionalnost se može smanjiti sa oko 4000 na oko 10 do 15 dimenzija bez značajnog gubitka informacije potrebnog za *k-nn* klasifikaciju.

3.2.4. *k-nn* klasifikacija

Očitanje registarskih pločica na mobilnoj platformi u ovom radu obavlja se pomoću *k-nn* postupka klasifikacije. Slika registarske pločice prvo se obrađuje postupcima opisanim u poglavlju 2 što kao rezultat daje slike znakova. Svakoj toj slici smanjuje se dimenzionalnost postupkom *PCA*. Zatim se svaka slika klasificira *k-nn* postupkom.

k-nn ili *k - najbližih susjeda* (algoritam 3) je algoritam koji se zasniva na traženju najbližih primjera iz skupa primjera za učenje. Neka matrica S predstavlja skup primjera za učenje. Neka je svaki primjer zapisan u jednom stupcu matrice. Svakom stupcu matrice dodijeljena je kategorija kojoj sadržani primjer pripada. Neka p označava primjer koji je potrebno klasificirati. Algoritam se svodi na računanje euklidske (ali može i neke druge) udaljenosti između primjera p i svakog primjera zapisanog u matrici S. Uzima se k najmanjih udaljenosti te se primjer p klasificira u kategoriju koja se najviše puta pojavila u skupu od k najmanjih udaljenosti. k je obično neparan broj.

U ovom radu skup primjera za učenje predstavlja matrica A koja je definirana u poglavlju 3.2.1 i nad kojom je obavljan *PCA* postupak. Kategorije dodijeljene stupcima matrice su u ovom slučaju skup svih brojki od 0 do 9 i velikih slova hrvatske abecede od A do Z bez slova s dijakritičkim znakovima ($\acute{C}$, $\check{C}$, $Đ$, $\check{S}$ i $\check{Z}$). Iako registarske pločice sadrže neka slova s dijakritičkim znakovima, njihovo zanemarivanje ne utječe na jednoznačnost očitanja. Primjer p je slika znaka dobi-

vena postupkom opisanom u poglavlju 2 smanjene dimenzionalnosti. Smanjenje dimenzionalnosti slike obavlja se množenjem već izračunatog vektora značajki W slikom zapisanom u obliku jednostupčane matrice. Slici p prethodno se oduzima već izračunata srednja slika $\overline{A}$. Time se dobila slika jednakog zapisa i dimenzionalnosti kakvu imaju slike iz skupa za učenje (matrice A).

Algoritam 3 Algoritam *k-nn*

1. Neka matrica A predstavlja skup za učenje smanjene dimenzionalnosti, jednostupčana matrica p sliku smanjene dimenzionalnosti koju treba klasificirati, a U neka predstavlja prazan skup.
2. Za svaki stupac matrice A napravi sljedeće:
   (a) Izračunaj euklidsku udaljenost između trenutnog stupca matrice A i slike p.
   (b) U skup U dodaj par (c, d) gdje je c kategorija pridružena trenutnom stupcu, a d euklidska udaljenost stupca od slike p.
3. Sortiraj skup U po udaljenosti d te iz skupa izbaci sve elementa osim prvih k elemenata.
4. Kategoriju slike p određuje kategorija c koja se u skupu U pojavljuje najviše puta.

Nakon što se ovaj algoritam provede za sve slike znakova s pločice, tekst pločice se ispisuje na zaslonu uređaja. Uspješnost algoritama ovisi o ulaznoj slici i o skupu za učenje. Za uspješnu evaluaciju treba postojati oko 30-ak primjera svakog znaka u skupu za učenje. Također, ulazna slika mora biti ispravno rotirana te bez šuma i nepravilnosti. U suprotnom rezultat evaluacije je obično netočan.

4. Implementacija

Programsko ostvarenje implementirano je oko jednostavne biblioteke posebno razvijene za ovaj projekt intuitivnog imena *SimpleCV*. Riječ je o biblioteci koja omogućava rad sa slikama, obavljanje nekih algoritama nad njima te koja implementira *PCA* postupak. Biblioteka ja napisana u jeziku *C++* što ju čini platformno neovisnom. Ostatak programske implementacije napisan je u jeziku *Objective-C* odnosno *Objective-C++*. To je nadskup jezika *C++* što znači da se unutar programa pisanog *Objective-C++* jezikom može koristiti i obični *C++* kod.

Aplikacijski dio ostvarenja koristi Appleovo nativno aplikacijsko programsko sučelje *Cocoa* za aplikacije koje se izvršavaju na stolnom računalu, odnosno *Cocoa Touch* za iPhone aplikaciju.

Ostvarenje je podijeljeno u nekoliko komponenata opisanih u nastavku.

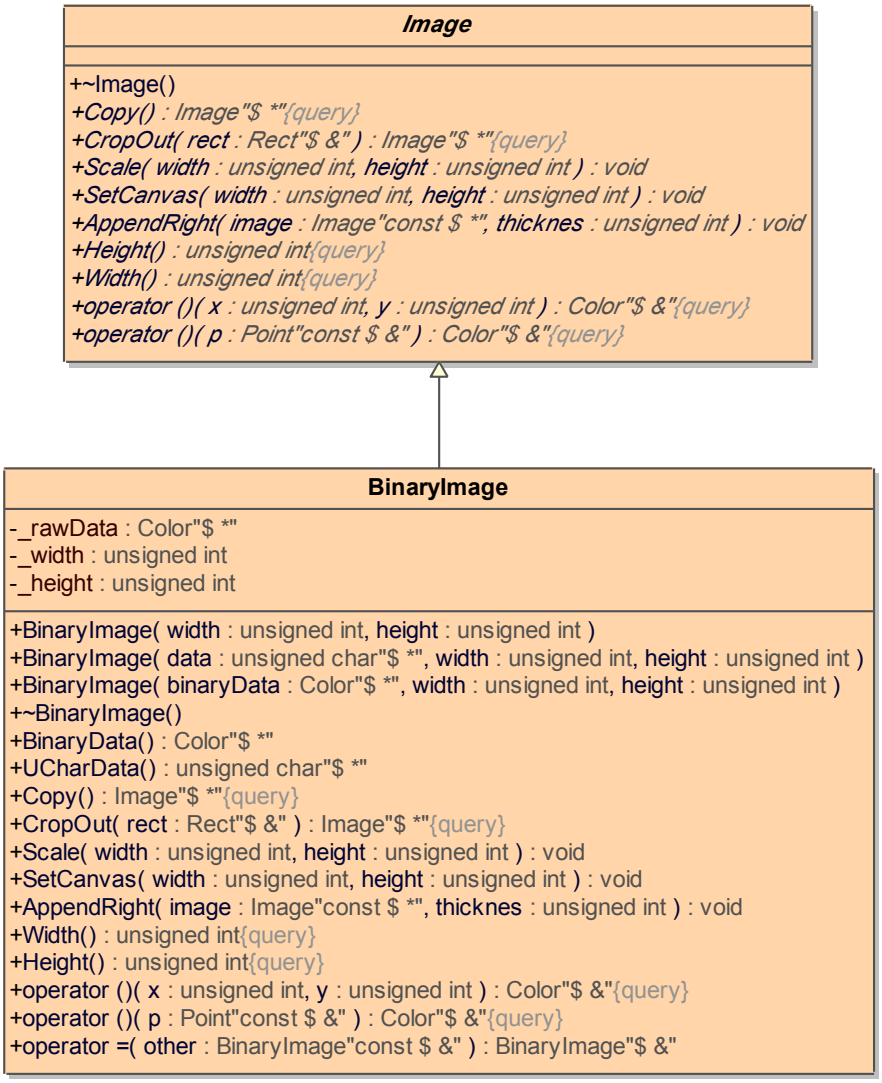
4.1. *SimpleCV*

Biblioteka *SimpleCV* olakšava rad sa slikama, omogućuje provedbu nekoliko algoritama korištenih u polju računalnog vida te implementira analizu svojstvenih komponenti nad skupom podataka. Sastoji se od nekoliko razreda podijeljenih u nekoliko grupa. Jedna od njih definira razred koji čuva sliku, druga matematičke strukture, treća algoritme te posljednja postupak analize svojstvenih komponenata.

4.1.1. Slika

SimpleCV omogućuje čuvanje slika. Definira apstraktno sučelje *Image* i njegove čiste virtualne metode. Podržane su najvažnije operacije obrade slika kao što je kopiranje, izrezivanje, promjena okvira i nadovezivanje slika. Detaljna implementacija prepuštena je izvedenim razredima.

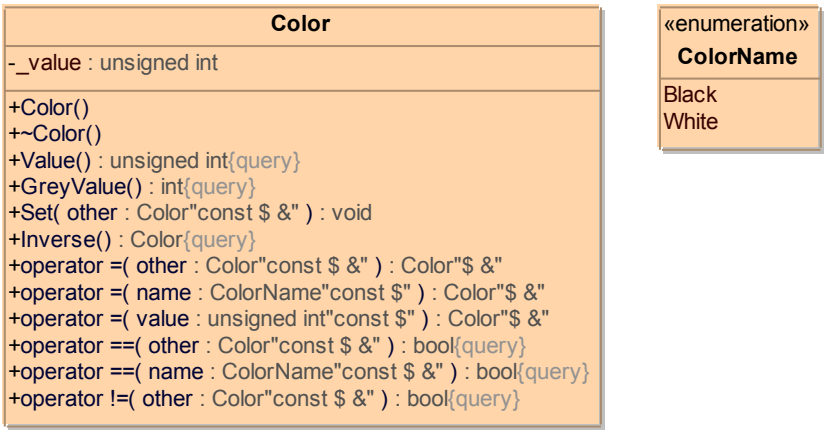
Obzirom da se u ovom radu barata samo sa binariziranim slikama, napravljena je samo jedna implementacija sučelja *Image*. Radi se o implementaciji binarizirane slike, dakle one koja može sadržavati samo crne i bijele slikovne elemente. Ime razreda je *BinaryImage*. *UML* dijagram obaju razreda prikazan je na slici 4.1.



Slika 4.1: Dijagram razreda za čuvanje slika

4.1.2. Boja

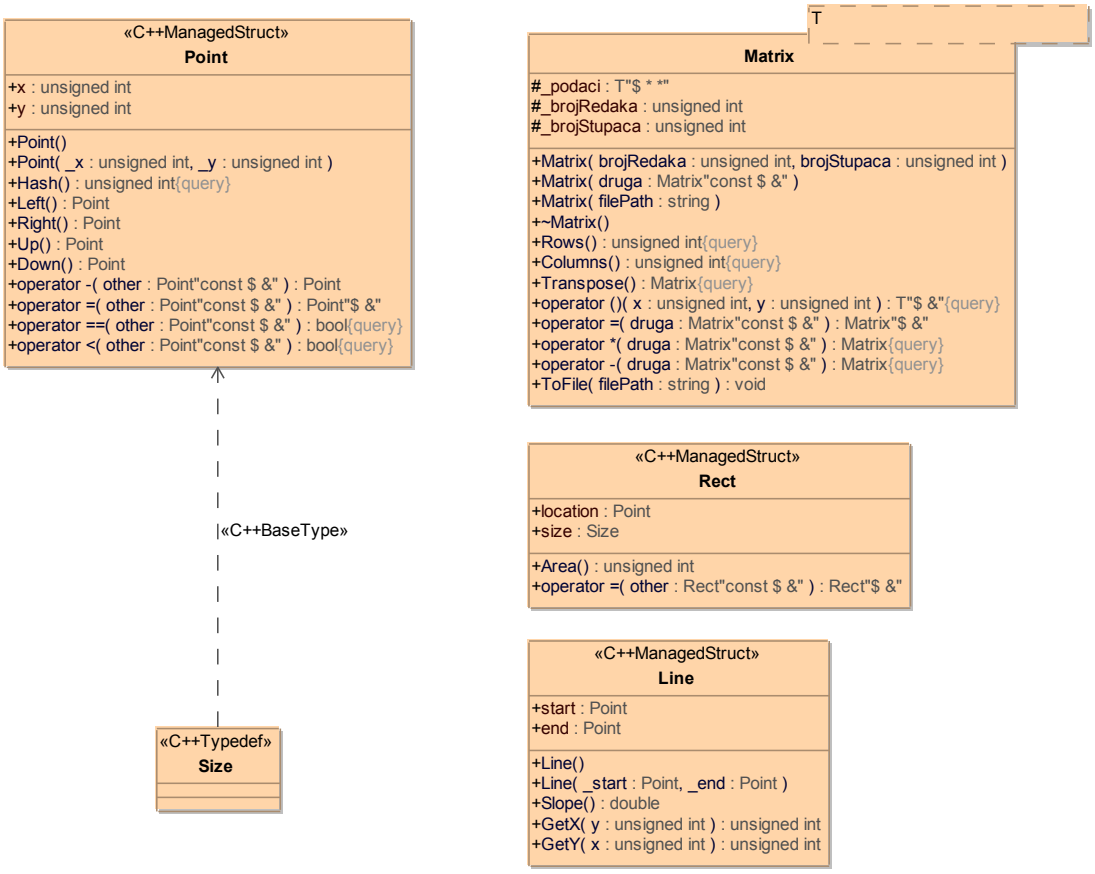
Gradivni elementi svake slike su *pikseli* određene boje. Binarna slika čuva sliku pomoću *niza boja* (*array*). Boja je implementirana konkretnim razredom *Color*. Implementacija razreda *Color* ne koristi polimorfizam što rezultira boljim performansama. Dijagram razreda *Color* i enumeracije *ColorName* koja omogućava pojednostavljeno korištenje čestih boja dan je na slici 4.2.



Slika 4.2: Dijagram razreda boja

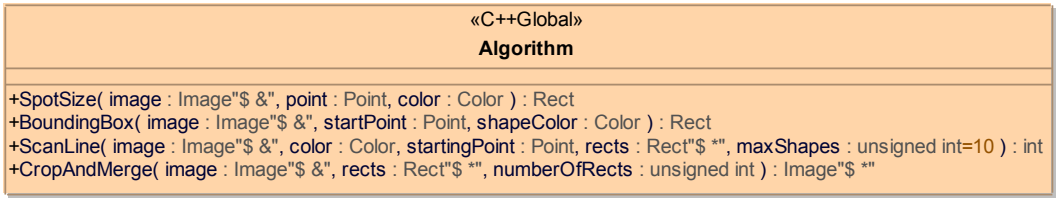
4.1.3. Matematičke strukture podataka

Algoritmi i postupak *PCA* implementirani bibliotekom *SimpleCV* prilikom računanja koriste određene matematičke konstrukcije stoga su one izdvojene u zasebne razrede. *Point* definira točku u 2D kartezijevom sustavu te nekoliko korisnih metoda i operacija vezanih uz točku. Razred *Matrix* predstavlja matricu te implementira metode i operacije potrebne za rad s matricama poput transponiranja, oduzimanja i matričnog množenja. Implementiran je kao parametrizirani objekt (*template*) čime se matrica ne ograničava na određeni tip podatka. Također implementira metodu koja omogućavam spremanje matrice u datoteku odnosno konstruktor za instanciranje matrice sa sadržajem iz datoteke. *Rect* predstavlja pravokutnik određen točkom i veličinom. *Line* predstavlja pravac određen dvjema točkama. Navedeni razredi prikazani su dijagramima na slici 4.3.



Slika 4.3: Matematičke strukture podataka

4.1.4. Algoritmi



Slika 4.4: Algoritmi

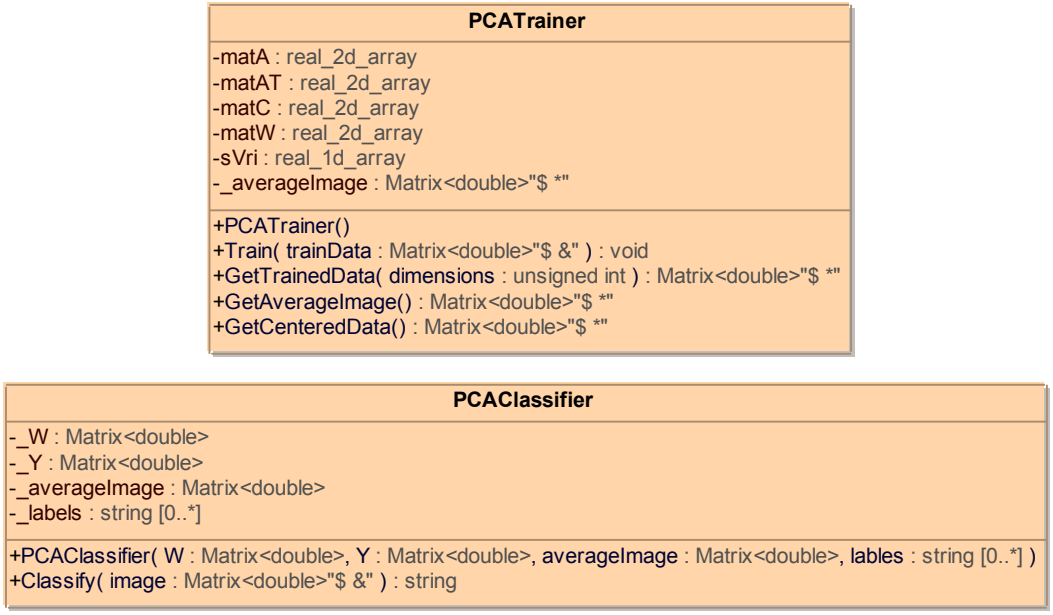
Algoritme predstavlja nekoliko funkcija. *SpotSize* računa okvir *mrlje* (otoka iste boje, objekta) u proslijeđenoj slici oko dane lokacije algoritmom *flood fill*. *BoundingBox* računa okvir prve desne *mrlje* odnosno objekta od točke *startingPoint* na kojeg naiđe algoritmom opisanim u poglavlju 2.3.2. Funkcija *ScanLine* predstavlja potpunu implementaciju algoritma za iščitavanje znakova čija je lokacija određena linijom. Algoritam je opisan u poglavlju 2.3.2. Funkcija implementira *pomicanje* po liniji te određivanje okvira znakova algoritmom koji obavlja funkcija *BoundingBox*. Posljedna funkcija, *CropAndMerge*, služi za izrezivanje

dijelova slike određenih poljem pravokutnika te spajanje tih djelova u novu sliku. Definicije ovih funkcija prikazane su na slici 4.4.

4.1.5. Analiza svojstevnih komponenata i klasifikacija

Implementacija *PCA* postupka podijeljena je u dva dijela – dio za učenje i dio za klasifikaciju. Razred *PCATrainer* predstavlja operacije koje je potrebno obaviti nad skupom podataka za učenje s ciljem smanjenja dimenzionalnosti. Glavni posao, dakle sva računanja, obavlja funkcija *Train*. Važno je napomenuti da izvršavanje ove funkcije nad većim skupom podataka traje jako dugo – u slučaju ulaznog skupa veličine oko $4000x700$ i do 45 minuta. *GetTrainedData* vraća skup reduciran na *dimensions* dimenzija.

PCAClassifier metodom *Classify* omogućava klasifikaciju novog primjera *image* primjenom *k-nn* metode. Razred prilikom instanciranja zahtjeva potrebne podatke za rad – reducirani skup za učenje Y, skup značajnih vektora W, srednju sliku *averageImage* te listu kategorija *labels*.

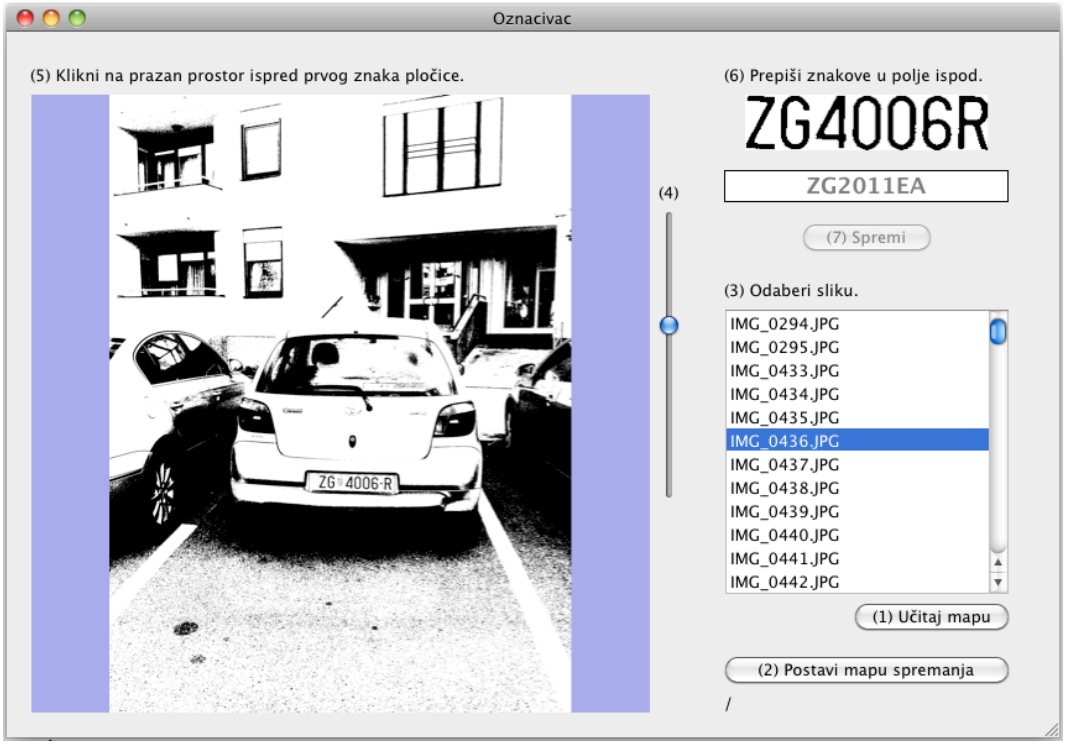


Slika 4.5: Dijagram PCA razreda

4.2. Aplikacije

4.2.1. *Označivač*

Označivač je aplikacija razvijena s ciljem olakšavanja obrade slika korištenih za izgradnju skupa za učenje. Aplikacija pruža intuitivno grafičko sučelje. Omogućava fluidnu obradu skupa slika, odnosno izvlačenje znakova s pločice. Izvlačenje se svodi na klik na prostor ispred prvog slova pločice, eventualnu prilagodbu binarizacije te prepisivanje teksta pločice. Aplikacija za svoj rad koristi biblioteku *SimpleCV*. Napisana je u *Objective-C* jeziku korištenjem *Cocoa API-ja* te stoga radi samo pod operacijskim sustavom *Mac OS*.



Slika 4.6: Označivač

Označivač svaku sliku znaka sprema u ime oblika *c_origName_n.png* gdje je *c* znak (slovo ili brojka), *origName* ime izvorne slike, a *n* redni broj znaka na pločici sa izvorne slike. Program također bilježi brag binarizacije svake slike u zasebnu datoteku.

4.2.2. *PCAImageProcessor*

PCAImageProcessor je aplikacija od nekoliko linija koda koja služi za generiranje reduciranog skupa za učenje. Aplikacija sve slike iz skupa za učenje učitava u

jednu matricu (matricu A) koju zatim prosljeđuje *PCATrainer* komponenti *SimpleCV* biblioteke. Obavlja se reduciranje dimenzionalnosti te se zatim rezultat sprema u datoteke – reduciranu matricu A, značajne vektore W i listu kategorija znakova. Ovaj aplikacija bi se čak moga nazivati i skriptom. Svi parametri radi jednostavnosti definirani su unutar koda.

4.2.3. *Čitač registarskih pločica*

Slika 4.7: Čitač registarskih pločica

Čitač registarskih pločica predstavlja krajnji proizvod ovog rada. Jezgra joj je *SimpleCV* biblioteka. Aplikacija omogućuje evaluaciju svih do sada opisanih postupaka i metoda te je primjer aplikacije računalnog vida na mobilnoj platformi. Sama aplikacija je vrlo jednostavne izvedbe. Sučelje se sastoji od dva dijela. Prvi dio omogućuje slikanje pločice, a drugi dio prikazuje pročitani tekst. Temelji se na arhitekturi *Model-Viewer-Controller*.

5. Ekspermentalni rezultati

Mobilna platforma zbog ograničenih resursa ne može pružiti performanse kao i *desktop* računala stoga je važno pisati što efikasnije aplikacije. Svi eksperimentalni rezultati prikazani u nastavku dobiveni su na platformi *Apple iPhone 4*.

5.1. Raspodjela procesorskog vremena

Zanimljive informacije pruža analiza dijelova sustava s obzirom na vremenski utrošak procesorskog vremena. Prosječan rezultat izvršavanja važnijih operacija izračunat na temelju nekoliko slijednih očitanja dan je u tablici 5.1. Mjerenje je vršeno na uređaju *iPhone 4* s procesorom *ARM Cortex-A8* i 512 *MB RAM* memorije te na *desktop* računalu s procesorom *Intel Core 2 Duo* i 4 *GB RAM* memorije. Slike slova i brojki kojima je obavljeno ispitivanje su veličine 32×32.

Operacija	ARM Cortex-A8 1.0 GHz	Intel Core 2 Duo 2.0 GHz
Binarizacija	630 ms	30 ms
Detekcija okvira	375 ms	45 ms
Segmentacija	7 ms	2 ms
Klasifikacija	205 ms	75 ms
Ukupno	1217 ms	152 ms

Tablica 5.1: Vremenski utrošak procesora

Uočava se da je najzahtjevnija operacija binarizacija slike. Razlog je rekurzivni algoritam određivanja praga binarizacije slike što bi se moglo ispraviti korištenjem nekog složenijeg, ali efikasnijeg algoritma. Iduća *skupa* operacija obavlja finu detekciju okvira pločice uporabom *flood fill* algoritma. Iako je u ovom radu implementirana najefikasnija varijanta tog algoritma, vrijeme njegovog izvođenja nije zanemarivo. Određivanje okvira svih znakova algoritmom opisanim u po-

glavlju 2.3.2 obavlja se vrlo brzo. Proces klasifikacije traje u prosjeku 200 *ms* no u njega je uključeno i skaliranje slika znakova na odgovarajuću veličinu. Također uočava se velika razlika u brzini izvođenja između mobilne i stolne platforme.

5.2. Točnost očitanja

U tablici 5.2 prikazani su rezultati očitanja za 15 slijednih primjera. Identičan test napravljen je za dvije konfiguracije *PCA* postupka – slikama znakova (u što su uključene slike znakova iz skupa za učenje te slika znaka koji se klasificira) skaliranih na veličine 32×32 i 24×24. Broj sačuvanih vektora značajki je 16.

	32 x 32		24x24	
Izvorni tekst	Pročitano	Točnost (%)	Pročitano	Točnost (%)
ZG8030Z	ZG8030Z	100	ZG8030Z	100
ZG2458DG	ZG24580G	87.5	ZG34580G	75
ZG8018EC	ZG8018EC	100	ZC8018EG	75
ZG0410MG	ZG0410MG	100	ZC04A0MG	75
ZG2622DJ	ZG2622DJ	100	ZG2C230J	62.5
ZG5843EC	ZG5843EC	100	ZC5B43EC	75
ZG5486DI	ZG5A68DI	87.5	ZC5A680I	37.5
ZG5235BK	ZG5235BK	100	ZG5335BK	87.5
ZG8042AK	ZG8042AK	100	ZG8043AK	87.5
ZG2884K	ZG2884K	100	ZG2884K	100
ZG6064K	ZG6064	85.7	ZG8D64K	75
ZG7512EI	ZG7512EI	100	ZGZ513EI	75
ZG7164AR	ZG7164AR	100	ZGZ164AB	75
ZG5861BF	ZG5861BF	100	ZG596ABE	62.5
ZG2118EG	ZG2118EG	100	ZC3118EC	75
Ukupan postotak točnosti		97.38		75.83

Tablica 5.2: Vremenski utrošak procesora

Vidljivo je da ispravnost prepoznavanja slova i brojki opada sa smanjenjem rezolucije. Također napravljeni su i testovi sa slikama veličina 16×16 i 64×64. Prve rezultiraju gotovo nikakvom ispravnošću prepoznavanja – točno se prepozna manje od 5% znakova. Slike veličine 64×64 daju gotovo identične rezultate očitanja kao slike veličine 32×32.

6. Zaključak

Ovaj rad prikazuje jednu od mnoštva primjena računalnog vida na mobilnim platformama. Pokazuje da se dosta složene operacije mogu bez problema obavljati na mobilnim procesorima što je dokaz dobrih performansi mobilne platforme. Iako se praktični dio ovog rada koncentrirao samo na Appleovu platformu, to ne znači da isti ne može biti jednako dobro ostvaren i na drugim platformama sličnih performansi. Očitanje registarskih pločica koristi nekoliko postupaka pomoću kojih dolazi do konačnog rješenja. Mnogi od tih postupaka implementiranih u ovom radu nisu jedini koji su se mogli primijeniti. Mnogi od njih nisu ni najefikasniji koji postoje no ipak obavljaju svoj posao vrlo zadovoljavajuće. Nepotrebno je govoriti da je to samo pokazatelj kako su mobilne platforme u zadnje vrijeme jako napredovale kad se govori o performansama i kvaliteti optičkih senzora.

Prilikom ostvarenja rada došlo je do nekoliko problema kao što su loša binarizacija ili neželjeni objekti prilikom segmentacije. Većina njih je ispravljena ili jednostavnim trikovima ili korištenjem drugih algoritama. U ovom radu opisani su postupci koji su davali najbolje rezultate i koji su pokazivali najbolju efikasnost izvođenja.

Ostvarenje rada uz evaluaciju mogućnosti mobilne platforme također daje i zadovoljavajuće rezultate očitavanja teksta te daje primjer novih mogućnosti primjene mobilnih uređaja.

Literatura

Neven Elezović. *Statistika i procesi*, 2009.

Lindsay Smith. *A tutorial on Principal Components Analysis*, 2002. URL http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf.

Ivana Sučić. *Primjena metode PCA nad skupom znakova*, 2009. URL http://www.zemris.fer.hr/~ssegvic/project/pubs/sucic09bs.pdf.

Wikipedia. *Flood fill*, 2011a. URL http://en.wikipedia.org/wiki/Flood_fill.

Wikipedia. *Grayscale*, 2011b. URL http://en.wikipedia.org/wiki/Grayscale).

Wikipedia. *Thresholding*, 2011c. URL http://en.wikipedia.org/wiki/Thresholding_(image_processing).

Strojno očitanje registarskih pločica na mobilnoj platformi

Sažetak

Povećanjem performansi mobilnih uređaja otvaraju se nove mogućnosti primjene računalnog vida. Ovaj rad bavi se problemom očitanja registarskih pločica korištenjem mobilne platforme. Opisuje postupke strojnog očitanja teksta te se osvrće na razne probleme koji prate strojno očitanje te njegovu implementaciju na mobilnoj platformi. Za potrebe rada napisana je jednostavna biblioteka koja olakšava rad sa slikama te implementira neke postupke obrade slika te postupke korištene u polju računalnog vida poput analize svojstvenih komponenti. U radu je prikazano nekoliko eksperimentalnih rezultata dobivenih ostvarenim sustavom na temelju kojih se može zaključiti da su se nove mobilne platforme dovoljno razvile za obradu složenih postupaka.

Ključne riječi: računalni vid, analiza svojstvenih komponenata, k-nn klasifikacija, mobilna platforma, registarske pločice

Vehicle registration plate recognition using a mobile platform

Abstract

As performance of mobile devices increases, new applications of computer vision are emerging. This work addresses the problem of vehicle plate recognition using a mobile platform. It describes the process of a machine recognition of a text and refers to the various problems that accompany it and its implementation on a mobile platform. For the purpose of this work, a simple library was written which facilitates a work with the images and implements some image processing algorithms and procedures used in the field of computer vision, such as principal component analysis. The paper presents several experimental results obtained from actual system on which it can be concluded that mobile platforms have developed enough to handle complex computing.

Keywords: computer vision, principal component analysis, k-nearest neighbor algorithm, mobile platform, vehicle registration plates