SVEUČILIŠTE U SPLITU

FAKULTET ELEKTROTEHNIKE, STROJARSTVA I BRODOGRADNJA

POSLIJEDIPLOMSKI STUDIJ ELEKTROTEHNIKE I INFORMACIJSKE TEHNOLOGIJE

Kvalifikacijski doktorski ispit

Klasifikacija dokumenata primjenom algoritama za strojno učenje na paralelnoj GPU arhitekturi

21.06.2014 Pjero Petej

Sadržaj

1. Uvod

U bilo kojem sustavu koji sadrži velike količine podataka, prije ili kasnije potrebno je naći neki način klasifikacije (kategorizacije) podataka. Količina podataka i informacija sa kojom se računalni sustavi danas susreću u stalnom je porastu što povlači da se se problemu klasifikacije podataka posvećuje sve više istraživanja.

U novije vrijeme pojavljuje se i pojam Big Data, koji se odnosi na izrazito velike i kompleksne količine podataka sa kojim više nije moguće upravljati na standardan način. Smatra se da se od 2012. godine, svakim danom se generira 2.5 eksabajtova podataka [1]. Uobičajene metodologije baratanja podacima (bazama podataka) i operacije nad istim (obrada, procesiranje, statistika...) više ne se nogu primjenjivati te se korisnici okreću novim metolodgijama.

Slični problemi se susreći i kod klasifikacije dokumenata, koji se mogu smatrati uobličenim podacima. Razvojem računala i digitalizacije, sve veći broj dokumenata (službenih i neslužbenih) postaje digitalno, čime se omogućuje bolja manipulacija istim. Neslužbeni dokumenti sve više postaju forme koji korisnici popunjavaju i spremaju na računala. Navedeno ubrzava razmjenu i obradu dokumenata.

Sa druge strane službeni dokumenti se najčešće unose u računalne sustave kao slike, gdje se nad istim vrše obrade, od jednostavnog prepoznavanja dokumenta do provjere autentičnosti. Također je bitno i spomenuti digitalne službene dokumente, gdje korisnici mogu digitalno potpisati dokument sa čime potvrđuju autentičnost. Digitalnih isprava je sve više, ali unatoč tome postoji veliki broj papirnatih dokumenata koji zbog svoje naravi moraju biti na fizičkom mediju (osobni dokumenti ili ovjereni dokumenti).

U ovom radu napravljen je pregled te su opisane metodologije koje se koriste u klasifikaciji skeniranih dokumenata te ekstrakciji značajki takvih dokumenata. Na polju računalnog vida i strojnog učenja napravljen je veliki napredak u prepoznavanju značajki pojedinih dokumenata, njihovoj ekstrakciji i daljnjoj obradi, ali zbog velikih količina podataka sa kojima se sustavi susreću i dalje se razvijaju nove metodologije, u pravilu brže i skalabilnije.

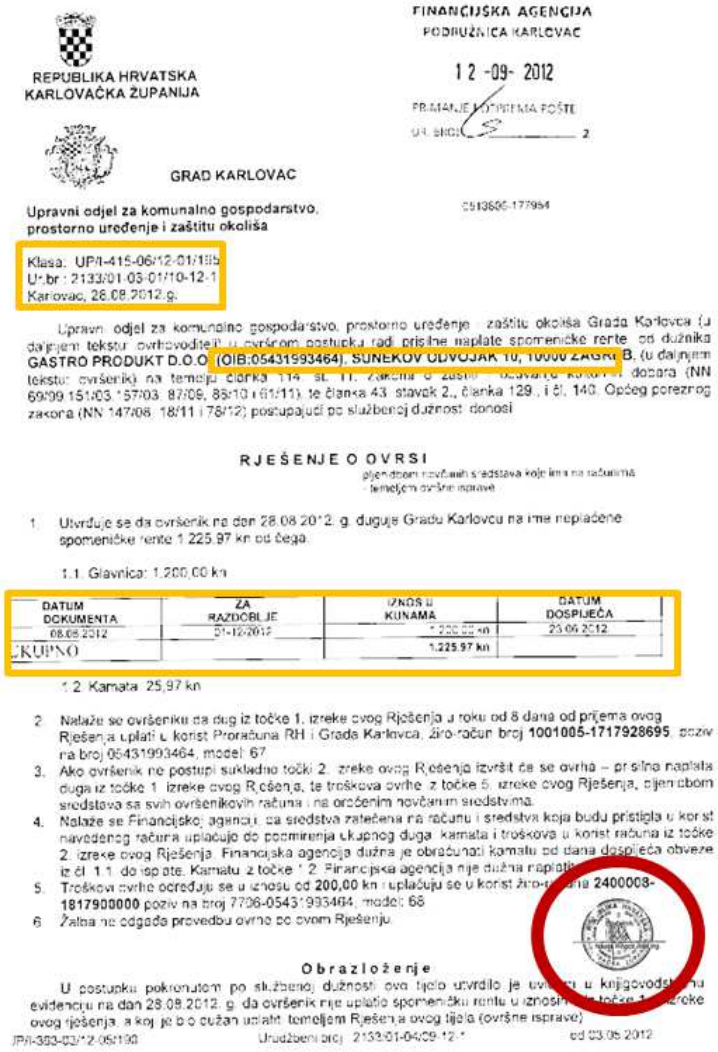
U prvim poglavljima opisani su uobičajeni problemi prilikom klasifikacije dokumenata i neke osnovne značajke dokumenta, a daljnja poglavlja opisuju distribuirane sustave koji imaju najveći problem klasifikacije velike količine dokumenata te GPU arhitektura kao potencijalno rješenje za efikasniju obradu istih.

1. Nadzirano i nenadzirano učenje

Nadzirano učenje je tip strojnog učenja gdje se ulazni skup podataka ispravno označava (na primjer klase), a rezultati izlaznog skupa, nakon učenja se uspoređuju sa oznakama iz ulaznog skupa. Na taj način promatra se proces učenja algoritma i podešavaju pojedini parametri algoritma sa ciljem veće točnosti.
Nenadzirano učenje se smatra težim zadatkom (barem u kontekstu klasifikacije) jer se ulazni skup predaju algoritmu za strojno učenje bez dodatnih informacija te algoritam sam grupira (kategorizira) ulazne podatke i prikazuje ih na izlazu. Proces grupacije se često puta naziva klasteriranjem (engl. clustering).
Kod klasifikacije dokumenta, ali i drugih problema klasifikacije, redovito se koriste kombinacije metoda nadziranog i nenadziranog učenja, a neke od njih objašnjene su u pogl. 4. Najčešći razlog korištenja više metoda je poboljšavanje performansi strojnog učenja i dobivanje boljih rezultata klasifikacije.
Pojedini algoritmi nenadziranog učenja mogu biti izrazito zahtjevni (na primjer algoritam k srednjih vrijednosti) za sustav, u smislu performansi, pa iako mogu imati zadovoljavajuće rezultate, nisu primjenjivi na manjim sustavima, a pogotovo u distribuiranim sustavima. Kompromis se može naći u optimiziranju podataka koje se predaju takvom algoritmu. Manja količina podatka osigurava manji broj računalnih operacija, a i dalje se koriste prednosti točnosti klasifikacije.
Algoritmima za strojno učenje (u obradi slike) se kao ulazni skup podataka redovito predaju značajke slike. Značajke i deskriptori značajke mogu biti različiti, od onih koji se baziraju na promjeni svjetline, do opisnih značajki. U svakom slučaju, područje detekcije i opisivanja značajki je još uvijek predmet istraživanja i redovito se definiraju nove modeli značajki, redovito brži i lakši u odnosu na prethodnike. Time se zapravo pridonosi "olakšavanju" ulaznog skupa podataka algoritmima za učenje. Neki od aktualnih modela značajki u obradi slike opisani su u pogl. 5.
Jedna od često korištenih metodologija jest da se dohvate značajke slike, zatim grupiraju u određene kategorije (nenadzirano učenje), a zatim tako grupirane značajke se predaju nadziranom algoritmu za strojno učenje, recimo stroju sa potpornim vektorima.

2. Problemi i izazovi klasifikacije dokumenata

Klasifikacija dokumenata se vrši na temelju pojedinih dijelova dokumenta koji jednoznačno određuju klasu dokumenta. Pojedina regija ili dio dokumenta koji je bitan za klasifikaciju uvelike ovisi o poslovnom procesu koji kreira dokument. U pravilu, ne postoji standardan način kreiranja pisanih službenih dokumenata, ali kod većine navedenih dokumenta postoje sličnosti (Slika 1).



REPUBLIKA HRVATSKA
KARLOVAČKA ŽUPANIJA

GRAD KARLOVAC

Upravni odjel za komunalno gospodarstvo, prostorno uređenje i zaštitu okoliša

FINANCIJSKA AGENCIJA
PODRUŽNICA KARLOVAC
12 -09- 2012

Klasa: UP/I-415-06/12-01/185
Ur.br: 2133/01-03-01/10-12-1
Karlovac, 26.08.2012.g.

GASTRO PRODUKT D.O.O. (OIB:05431993464), SUNEKOV ODVOJAK 10, 10000 ZAGREB

RJEŠENJE O OVRSI

DATUM DOKUMENTA	ZA RAZDOBLJE	IZNOS U KUNAMA	DATUM DOSPIJEĆA
08.08.2012	01-12/2012	1.200,00 kn	23.08.2012
UKUPNO		1.225,97 kn	

1. Utvrđuje se da ovršenik na dan 28.08.2012. g. duguje Gradu Karlovcu na ime neplaćene spomeničke rente 1.225,97 kn od čega:

1.1. Glavnica: 1.200,00 kn

1.2. Kamata: 25,97 kn

2. Nalaže se ovršeniku da dug iz točke 1. izreke ovog Rješenja u roku od 8 dana od prijema ovog Rješenja uplati u korist Proračuna RH i Grada Karlovca, žiro-račun broj **1001005-1717928695**, poziv na broj 05431993464, model: 67
3. Ako ovršenik ne postupi sukladno točki 2. izreke ovog Rješenja izvršit će se ovrha – prisilna naplata duga iz točke 1. izreke ovog Rješenja, te troškova ovrhe iz točke 5. izreke ovog Rješenja, plijenidbom sredstava sa svih ovršenikovih računa i na drugim novčanim sredstvima.
4. Nalaže se Financijskoj agenciji da sredstva zatečena na računu i sredstva koja budu pristigla u korist navedenog računa uplaćuju do podmirenja ukupnog duga: kamata i troškova u korist računa iz točke 2. izreke ovog Rješenja. Financijska agencija dužna je obračunati kamatu od dana dospijeća obveze iz čl. 1.1. do isplate. Kamatu iz točke 1.2. Financijska agencija nije dužna naplatiti.
5. Troškovi ovrhe određuju se u iznosu od **200,00** kn i uplaćuju se u korist žiro-računa **2400008-1817900000** poziv na broj 7706-05431993464, model: 68
6. Žalba ne odgađa provedbu ovrhe po ovom Rješenju.

Obrazloženje

U postupku pokrenutom po službenoj dužnosti ovo tijelo utvrdilo je ovršenika u knjigovodstvenoj evidenciji na dan 28.08.2012. g. da ovršenik nije uplatio spomeničku rentu u iznosu iz točke 1. izreke ovog rješenja, a koji je bio dužan uplatiti temeljem Rješenja ovog tijela (ovršne isprave)

UP/I-363-03/12-05/190 Urudžbeni broj: 2133-01-04/09-12-1 od 03.09.2012

Slika 1 - interesne regije dokumenta

Recimo, moguće je razlikovati zaglavlje, tijelo i podnožje dokumenta. U većini slučajeva, potpis i pečat će se nalaziti u podnožju dokumenta, podaci o pošiljatelju (autoru) u zaglavlju, a podaci o sadržaju dokumenta u tijelu dokumenta.

Razlikovanje navedenih područja i definiranje strogih granica je poprilično težak problem. Čak i kod dokumenata istog autora, postoje značajne razlike u definiranu

regija. Zato je potrebno definirati regije, i u prvoj fazi, korištenjem nekgo od algoritama strojnog učenja, dobiti procjenu granica regije.

2.1 Definiranje predloška

Prilikom obrade dokumenta potrebno je definirati predložak koji definira određene regije interesa za klasifikaciju. Pojedini pristupi u definiranju regije uključuju segmentaciju slike ([2],[3]) te treniranjem na velikom uzorku dokumenata iste klase. Ovakav pristup se može smatrati strojnim učenjem bez nadzora (, gdje se algoritmu za učenje omogući da na temelju segmenata slike odlučuje gdje su granična područja i samim time granice regije.

Drugi pristup bi bio nadzirano učenje, gdje se algoritmu za strojno učenje u trening fazi (na postojećem uzorku klasa dokumenta) učenja označe područja razgraničenja regije. Nakon trening faze, granice regija se strogo definiraju te se sa testnim uzorkom ispituje rad klasifikatora.

Prednost drugog pristupa je u boljim granicama regija i općenito manjem trening uzorku, ali i stvaranju veće greške kod novih dokumenata koji se razlikuju od onih u trening fazi.

Za ispravan ishod klasifikacije potrebno je pravilno implementirati trening i test fazu algoritma za strojno učenje. Neovisno o tipu algoritma, može se smatrati da su ove dvije faze ključne za definiranje pravilnog ishoda klasifikacije. Trening faza algoritma uključuje poznati skup podataka koji se predaje stroju za učenje sa ciljem definiranja modela za buduće prepoznavanje. Jedan od najstarijih algoritama za strojno učenje jest

3. Pregled algoritama za strojno učenje (klasifikacija slike)

Područje strojnog učenja se eksponencijalno razvija u posljednjih 30-ak godina. Algoritmi za strojno učenje mogu se svrstati u dvije grupe: algoritmi za nadzirano strojno učenje i algoritmi za strojno učenje bez nadziranja. Iako je broj algoritama velik, u ovom radu su obrađeni klasifikatori najčešće korišteni u klasifikacije slike [12] (Naivni Bayes-ov, Stroj sa potpornim vektorima, Neuronske mreže, Algoritam k srednjih vrijednosti). Također prikazan je i relativno noviji (2012.) algoritam koji ima odlične rezultate klasifikacije (Random fern).

3.1 Naivni i Polu-navini Bayes klasifikator

Bayes-ova vjerojatnost, često nazivana i aposteriori vjerojatnost, jest ona vjerojatnost koja opisuje kako nova saznanja o sustavu utječu na ishod vjerojatnosti cijelog sustava. Bayes-ovu vjerojatnost moguće je prikazati preko uvjetovanih događaja. Pretpostavimo da postoje događaji A i B. Tada vjerojatnost da se dogodio događaj A uz uvjet događaja B možemo zapisati kao:

$$P(A|B) = \frac{P(A \cap B)}{P(B)}, uz\ P(B) \neq 0 \quad (1)$$

Isto tako, vjerojatnost da se dogodio događaj B uz uvjet događaja A jest

$$P(B|A) = \frac{P(A \cap B)}{P(A)}, uz\ P(A) \neq 0 \quad (2)$$

Iz (1) i (2) moguće je zapisati sljedeće:

$$P(B|A)P(A) = P(A \cap B) = P(A|B)P(B) \quad (3)$$

Iz (3) slijedi:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}, uz\ P(B) \neq 0 \quad (4)$$

Izraz (4) je tzv. Bayes-ova formula i predstavlja temelj Bayes-ove vjerojatnosti.

Naivni Bayes klasifikator je vjerojatnosni klasifikator koji se temelji na Bayes-ovom pravilu, a pretpostavlja međusobnu nezavisnost značajki sustava. Na primjer, možemo smatrati da objekt pripada nekoj klasi ako ima određen značajke, uz pretpostavku da su te značajke nezavisne (objekt možemo smatrati narančom ako je okrugao, narančast, ima određenu masu, gustoću itd). Značajke sustava, to jest njihova međusobna nezavisnost, ovise o sustavu, a pretpostavka o nezavisnim značajkama je najčešće pogrešna. Unatoč tome pokazala se dovoljno dobra kod jednodimenzionalnih problema [13].

Ako se želi izračunati najveća aposteriori vjerojatnost da neka klasa ima određene značajke, može se koristiti izraz (5):

$$\underset{k}{argmax}\ P(C_k|f_1, f_2 \ldots . f_N) \quad (5)$$

Prema Bayes-ovom pravilu, izraz (5) se može izračunati preko izraza 6) (budući da je nazivnik u izrazu (4) konstantan):

$$\underset{k}{argmax}\, P(f_1, f_2 \ldots . f_N|C_k)\, P(C_k) \tag{6}$$

No, prema izrazu (6) potrebno je računati vjerojatnosti da pojedina značajka pripada pojedinoj klasi (te ako su značajke međuzavisne i njihove međusobne vjerojatnosti). Takav račun je na većem sustavu je izuzetno kompliciran te u praksi nije primjenjiv.
Iz gore navedenih razloga, naivni Bayes klasifikator pretpostavlja međusobnu nezavisnost značajki te bi se tada dio izraza (6) mogao računati kao:

$$P(f_1, f_2 \ldots . f_N|C_k) = \prod_{i=1}^{N} P(f_i|C_k) \tag{7}$$

Koristeći izraz (7), izraz (5) moguće je zapisati kao:

$$\underset{k}{argmax}\, P(C_k) \prod_{i=1}^{N} P(f_i|C_k) \tag{8}$$

3.2 Stroj sa potpornim vektorima

Stroj s potpornim vektorima (engl. *Support Vector Machine)* je binarni klasifikator koji konstrukcijom hiperravnine u više-dimenzionalnom prostoru stvara model koji predviđa kojem od dva razreda pripada novi uzorak.

Linearno odvojivi razredi

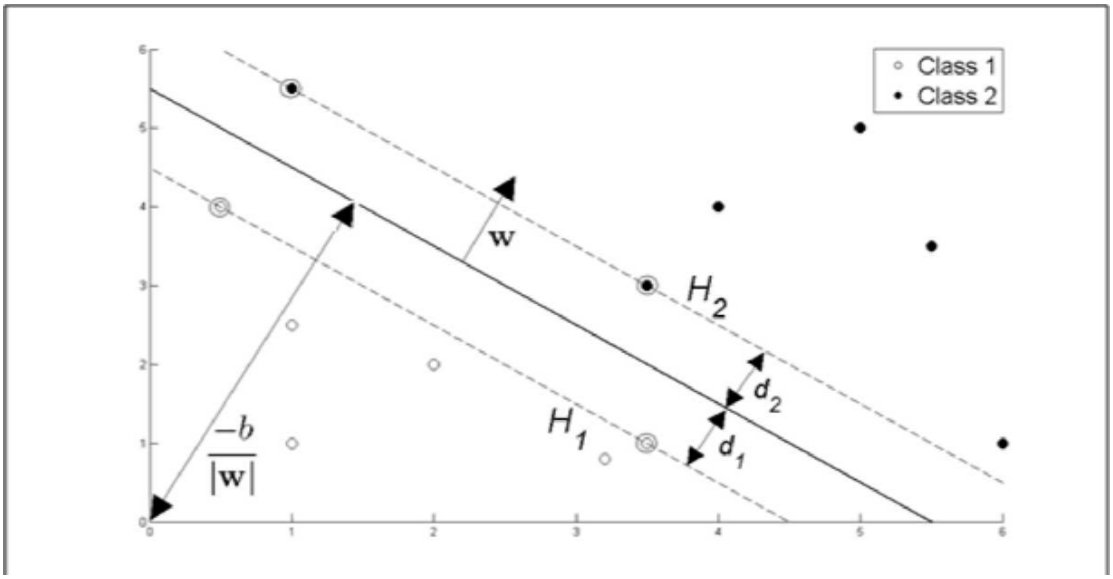
U skupu za učenje imamo *L* vektora (točaka u D-dimenzionalnom prostoru), gdje svaki uzorak ima D atributa, odnosno komponenti vektora te pripada jednom od dva razreda $y_i = -1$ ili 1. Oblik jednog ulaznog podatka prikazan je izrazom (9) [14].

$$\{x_i, y_i\} \text{ gdje je } i = 1 \ldots L,\ y_i \in \{-1,1\},\ x \in R^D \tag{9}$$

Pretpostavlja se da su podaci linearno razdvojivi, što znači da možemo nacrtati pravac u koordinatnom sustavu s osimax_1i $x_2$za slučaj $D = 2$, odnosno hiperravninu za slučaj D > 2. Hiperravninu opisujemo izrazom$w \cdot x + b = 0$, gdje je:

- *w* normala hiperravnine
- $\frac{b}{\|w\|}$okomita udaljenost hiperravnine od ishodišta koordinatnog sustava

Potporni vektori su uzorci najbliži razdvajajućoj hiperravnini i zato se najteže klasificiraju, a cilj stroja s potpornim vektorima jest da odabere hiperravninu maksimalno udaljenu od najbližih uzoraka oba razreda. Grafički prikaz jednostavnog dvodimenzionalnog slučaja prikazan je na *Slika 2.*



Slika 2 - Određivanje potpornih vektora

Implementacija stroja s potpornim vektorima sada se svodi na odabir parametara *w* i *b*, takvih da ulazne podatke možemo opisati sljedećim izrazima:

$$x_i \cdot w + b \geq +1 \; za \; y_i = +1 \tag{10}$$

$$x_i \cdot w + b \leq -1 \; za \; y_i = -1 \tag{11}$$

Kombinacijom ta dva izraza dobivamo

$$y_i(x_i \cdot w + b) - 1 \geq 0, \forall i \tag{12}$$

Ravnine H_1 i H_2 na kojima leže potporni vektori (označeni kružićima na *Slika 2*) možemo opisati sljedećim izrazima

$$x_i \cdot w + b = +1 \; za \; H_1 \tag{12}$$

$$x_i \cdot w + b = -1 \; za \; H_2 \tag{13}$$

Definiramo vrijednosti d_1 i d_2 kao udaljenosti od H_1 i H_2 do hiperravnine. Ekvidistantnost hiperravnine od H_1 i H_2 podrazumijeva $d_1 = d_2 = \frac{1}{\|w\|}$. Tu vrijednost nazivamo marginom stroja s potpornim vektorima. Ako želimo odabrati hiperravninu maksimalno udaljenu od potpornih vektora, moramo maksimizirati marginu, što je ekvivalentno pronalaženju:

$$min\|w\| \text{ takav da } y_i(x_i \cdot w + b) - 1 \geq 0, \forall i \tag{13}$$

Linearno neodvojivi razredi

Ako želimo proširiti funkcionalnost Stroja s potpornim vektorima na linearno neodvojive razrede, potrebno je ublažiti uvjete (8) i (9) uvođenjem nenegativne vrijednosti ξ_i:

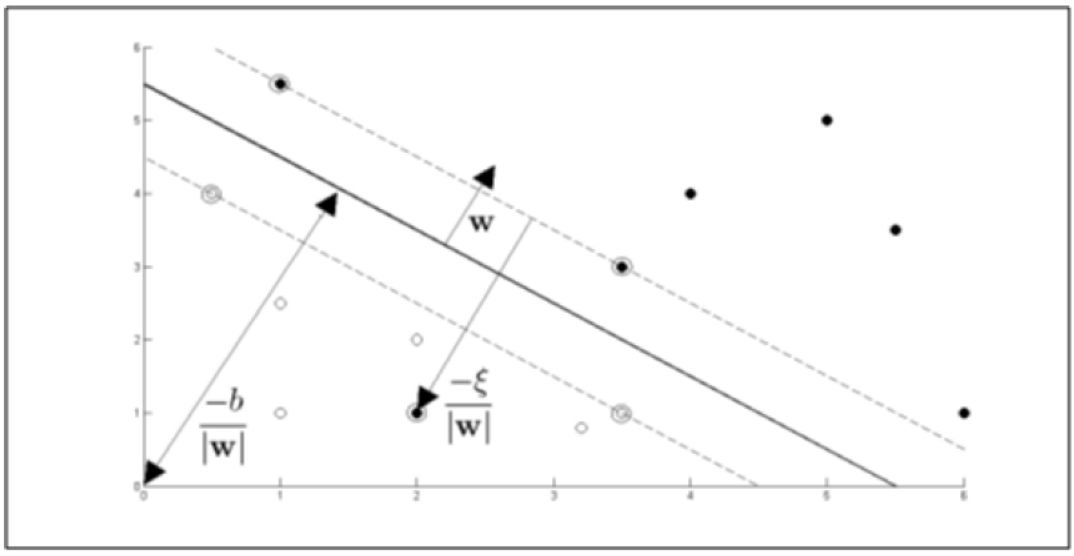
$$x_i \cdot w + b \geq +1 - \xi_i \; za \; y_i = +1 \tag{14}$$

$$x_i \cdot w + b \leq -1 + \xi_i \; za \; y_i = -1 \tag{15}$$

Kombinacijom ta dva izraza dobivamo:

$$y_i(x_i \cdot w + b) - 1 + \xi_i \geq 0, \;\; \xi_i \geq 0 \;\; \forall i \tag{16}$$

Ova metoda naziva se metoda meke margine (engl. *soft margin method* [15]), a izvorno je nastala s idejom dozvoljavanja pogrešnog označavanja razreda prije samog postupka učenja. Na *Slika 3* prikazana je hiperravnina (pravac) kroz dva linearno neodvojiva razreda. Vidljiv je i uzorak s pogrešne strane hiperravnine zbog kojeg prostor nije linearno odvojiv. Mjera udaljenosti tog uzorka od pripadajućeg potpornog vektora jest ξ.



Slika 3 - linearno neodvojiv razredi

Sada se odabir razdvajajuće hiperravnine svodi na pronalaženje:

$$min \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{L} \xi_i \;\; \text{takav da} \;\; y_i(x_i \cdot w + b) - 1 + \;\; \xi_i \geq 0, \forall i \tag{17}$$

Vrijednost *C* u izrazu (17) predstavlja faktor pogreške, kojim dozvoljavamo određene pogreške pri treniranju, bez čega pronalazak hiperravnine ne bi bio moguć.

Jezgreni trik

Do sada opisani (linearni) klasifikator zapravo se naziva Klasifikator optimalne granice (*engl*. *MaximumMarginClassifier*). Stroj s potpornim vektorima je poopćeni Klasifikator optimalne granice za nelinearnu klasifikaciju, što se postiže postupkom poznatim pod nazivom Jezgreni trik (engl. *Kernel Trick*).

Osnovna ideja jest u izrazu (17) zamijeniti ulazni vektor značajki x_i s funkcijom$\phi(x_i)$koja ulazni vektor preslikava iz *n*-dimenzionalnog u *m*-dimenzionalni prostor, uz m>>n. U novom, *m*-dimenzionalnom, prostoru su uzorci linearno odvojivi. Problem predstavlja računanje unutarnjeg produkta vektora $\phi(x_i)$i *w* jer je nova dimenzionalnost puno veća, ponekad i beskonačna. Međutim, moguće je napisati Jezgrenu funkciju (engl. *Kernel function)* $K(x_i, x_j) = \phi(x_i)^T \cdot \phi(x_j)$ koju je moguće izračunati puno jednostavnije nego eksplicitno računati skalarni produkt $\phi(x_i)^T \cdot \phi(x_j)$.

Postoji nekoliko standardnih oblika Jezgrenih funkcija:

Linearna

$$K(x_i, x_j) = {x_i}^T \cdot x_j \tag{18}$$

Polinomna

$$K(x_i, x_j) = ({x_i}^T \cdot x_j + a)^b \tag{19}$$

Gaussova(RBF, engl. *RadialBasisFunction)*

$$K(x_i, x_j) = e^{-\gamma \|x_i - x_j\|^2}, \ \gamma > 0 \tag{20}$$

Gaussova jezgra je jezgra s najširom primjenom. Ona odgovara preslikavanju u beskonačno-dimenzionalni prostor. Sve što je linearno razdvojivo u početnom prostoru značajki, razdvojivo je i u prostoru određenom ovom jezgrom. Parametar određuje širinu „zvona" Gaussove krivulje.

Racionalna kvadratna [14]

$$K(x_i, x_j) = 1 - \frac{\|x_i - x_j\|^2}{\|x_i - x_j\|^2 + \tau} \tag{21}$$

Gornja jezgra je dobra zamjena za Gaussovu jezgru ukoliko je potrebno izbjeći potenciranje.

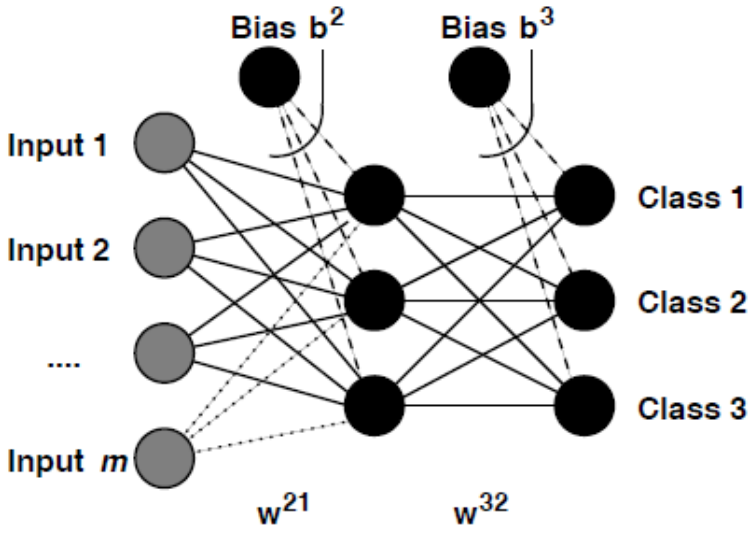
Sigmoidalna

$$K(x_i, x_j) = tanh\ (\kappa {x_i}^T \cdot x_j + a) \tag{22}$$

Ključ dobre klasifikacije strojem s potpornim vektorima leži u odabiru parametara jezgrene funkcije i ranije spomenutog parametra *C* - faktora pogreške.

3.3 Umjetne neuronske mreže

Umjetna neuronska mreža (*engl. Artificial Neural Networks* – ANN) je skup umjetnih neurona (najčešće kao apstraktnih pojmova) koji su međusobno povezani i interaktivni kroz operacije obrade signala. Mreža može imati niz ili jedan ulaz ali uvijek jedan izlaz, između kojih se nalazi jedan ili više tzv. skrivenih slojeva (tzv. višeslojne mreže). Pojedinačni neuroni su, kao i slojevi, međusobno spojeni vezama kroz koje idu signali. Veze među njima se aktiviraju ako je zadovoljen uvjet postavljen aktivacijskom funkcijom [7].
Umjetne neuronske mreže su često korišten alat u obradi slike, prvenstveno radi svoje visoke razine nelinearnosti. ANN model je stvaran po uzoru na neuronske mreže u mozgu i iako postoji više modela ANN-a (Hopfield-ova mreža [6] ili Boltzmann-ov stroj [7]), ovdje će biti obrađen model koji se najčešće koristi u obradi slike, a to je *engl. feed forward* ANN (ANN gdje se podaci šalju samo sljedećem sloju). Grafički prikaz spomenute mreže dan je na Slika 4:



Slika 4 - Primjer neuronske mreže

Sustav koji opisuje neuronsku mrežu najčešće opisuje sljedeće:

- Sadrži veliki broj identičnih obradnih jedinica

- Postoji veza između navedenih jedinica
- Jedinice obrađuju podesive parametre (težinske funkcije) koji definiraju funkciju sustava
- Ne postoji sustav nadzora nad pojedinim težinskim funkcijama

Feed Forward ANN [9] ili mreže sa višeslojnim perceptronima (*engl. multi-layer perceptrons* - MLP) se sastoje od međusobno povezanih slojeva jedinica obrade ili neurona (Slika 4)
Na Slika 4 korištene su sljedeće oznake: težinske funkcije između slojeva *p* i *q* označena je w^{qp}; pomak, ulazni i izlazni vektori sloja označeni su sa b^p, I^p i O^p. Općenito feed forward ANN je parametrizirana, adaptivna vektorska funkcija koju je moguće istrenirati za izvođenje klasifikacije ili regresije. Klasifikacijska feed forward umjetna mreža vrši preslikavanje:

$$N: \mathbb{R}^d \to \langle r_{min}, r_{max} \rangle^m \tag{23}$$

gdje je *d* dimenzija ulazno prostora (značajki), *m* je broj različitih klasa i $\langle r_{min}, r_{max} \rangle$ je opseg svake izlazne jedince. Sljedeća feed forward umjetna neuronska mreža sa jednim skrivenim slojem može ostvariti spomenuto preslikavanje:

$$N(x; W, B) = f(w^{32^T} f(w^{21^T} x - b^2) - b^3) \tag{24}$$

Wje skup težinskih funkcija, koji sadrži matricu težinskih funkcija koje povezuju ulazni sloj i skriveni sloj (w^{21}) i vektor koji spaja skriveni sloj sa izlaznim slojem (w^{32}); $B(b^2\ i\ b^3)$ sadrži podatke pomaka skrivenog i izlaznog sloja . Funkcija $f(a)$je nelinearna aktivacijska funkcija sa opsegom $\langle r_{min}, r_{max} \rangle$, koja se izvršava nad svakim elementom ulaznog vektora. Najčešće se koristi sigmoidalna funkcija $f(a) = \frac{1}{1+ e^{-a}}$ sa rasponom $\langle r_{min} = 0, r_{max} = 1 \rangle$; dvostruka sigmoidalna funkcija $f(a) = \frac{2}{1+ e^{-a}}$ili hiperbolna tangens funkcija $f(a) = \tanh(a)$, obje sa opsegom $\langle r_{min} = -1, r_{max} = 1 \rangle$.

Za klasifikaciju, ANN računa poseteriori vjerojatnosti zadanih vektora $x, P(w_j|x)$ gdje je w_j oznaka klase $j, j = 1, \dots m$. Klasifikacije se nadalje izvodi dodjeljivanjem ulaznog elementa x klasi koja ima najveću vjerojatnost. Feed forward ANN može se trenirati i na nadzirani način (u svrhu klasifikacije). U tom slučaju potrebno je uvesti trening uzorak $\mathcal{L} = \{(x, t)\}$, gdje je t_1 visoke vrijednosti (npr. 0.9) i t_k niske vrijednosti (npr. 0.1), $\forall k \neq l$. Trening algoritam, na primjer povratni propagacijski algoritam [10] tada minimizira funkciju kvadratnu greške:

$$E(W, B) = \frac{1}{2|\mathcal{L}|} \sum_{(x^t,\, t^1) \in \mathcal{L}} \sum_{k=1}^{c} (N(x^i; W, B)_k - t_k^i)^2 \tag{25}$$

Podešavanjem težinskih faktora i faktora pomaka feed forward umjetne neuronske mreže, sa dovoljno čvorova u skrivenom sloju i sa beskonačno veliki trening uzorkom, opisuju Bayes-ovu posteriori vjerojatnost [11]:

$$P(\omega_j|x) = \frac{P(\omega_j)p(x|\omega_j)}{p(x)}, j = 1, \dots, m \tag{26}$$

gdje je $P(\omega_j)$ apriori vjerojatnost klase j, $p(x|\omega_j)$ je funkcija gustoće uvjetovane vjerojatnosti klase j i $p(x)$ je promatrana vjerojatnost za x.

3.4 K-Mean Grupiranje

K-Mean algoritam je algoritam za pronalaženje grupa (klastera) te spada u algoritme nenadziranog učenja. Drugim riječima, računalu nije dan neki uzorak koji definira ponašanje algoritma nego računalo na temelju procjena algoritma u svakom koraku stvara klastere.
Osnovna ideja je inicijalno definirati broj klastera k i za svaki klaster formirati centroid tog klastera. Zatim svakoj točki promatranog skupa podataka dodijeliti pripadnost pojedinom klasteru prema najmanjoj apsolutnoj udaljenosti od pojedinog centroida. Nakon novoformiranih klastera, računa se novi centroid klastera. U svakoj sljedećoj iteraciji centroid svake klase se mijenja, u odnosu na novo stanje klastera. Algoritam se zaustavlja kada se centroidima prestane mijenjati položaj.K-Means algoritam minimizira funkciju cilja, u ovom slučaju funkciju kvadrata pogreške.

Funkcija:

$$d = \sum_{j-1}^{m} \sum_{i-1}^{n} \left| X_i^{(j)} - C_j \right|^2 \tag{27}$$

je odabrana za izračunavanje udaljenosti uzoraka i centroida. Predstavlja indikator udaljenosti za *n* uzoraka od njihovih centroida.

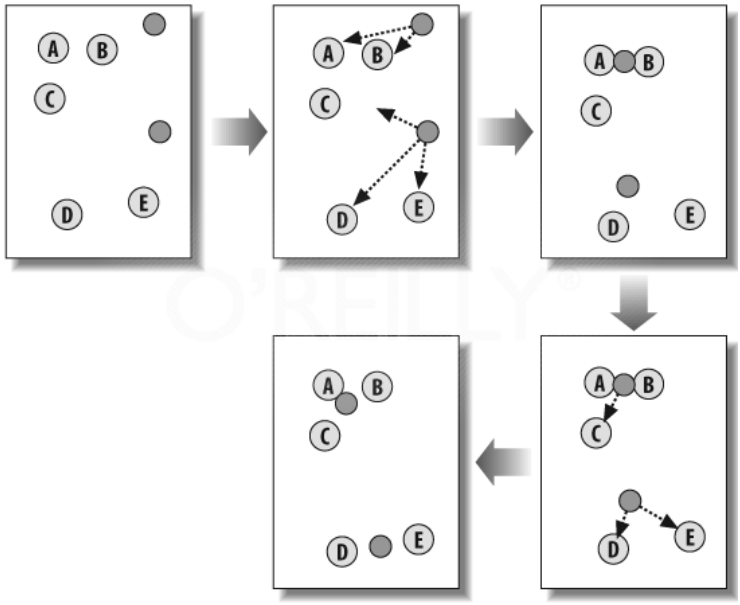
Algoritam sadrži sljedeće korake:

1. Odabir *K* točaka u prostoru zadanih uzoraka - ove točke su inicijalna grupa centroida.
2. Pridruživanje svakog uzorka grupi sa najbližim centroidom.
3. Kada su svi uzorci pridruženi, ponovno izračunavanje pozicija za *K* centroida.
4. Ponavljati korake 2 i 3 sve dok se centroidi ne prestanu pomicati.

Premda se može pokazati da K-Means procedura uvijek "uspješno" završava, ipak ovaj algoritam nužno ne pronalazi optimalnu konfiguraciju klastera. Algoritam je iznimno osjetljiv na početni slučajni odabir centara klastera. Da bi se ovaj učinak umanjio, K-Means algoritam se uobičajeno pokreće više puta.

K-Means je jednostavan algoritam koji se može primijeniti na različitim područjima. Kao i svaki algoritam, K-Means ima određene nedostatke. Možda najveći nedostatak jest to što nije specificiran način kako odabrati inicijalne položaje centroida. Popularan načinje "slučajno" odabrati *k* od zadanih uzoraka. Rezultat značajno ovisi o inicijalnom odabiru pozicije centroida te o vrijednosti *k.* Rješenje spomenute ovisnost moguće je naći je u većem broju pokušaja sa različitim odabirom pozicija inicijalnih centroida. Ipak, najveći problem je u tome što često unaprijed nije poznato koliko postoji klastera [16].

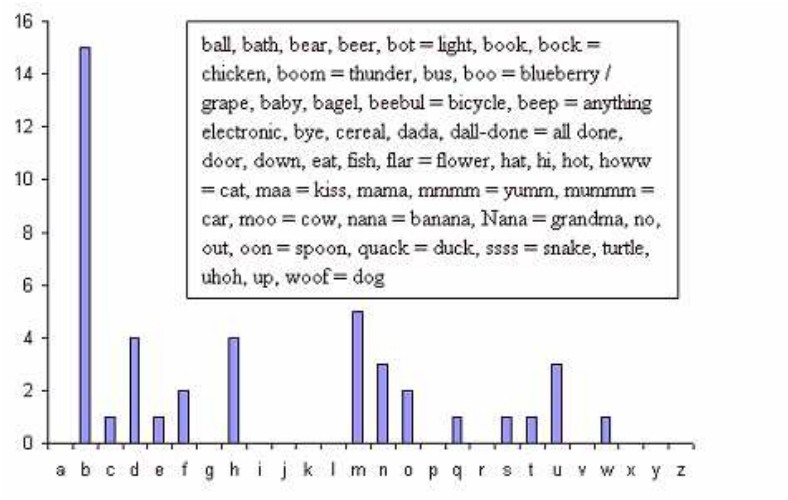
Ne postoji opće teorijsko rješenje za pronalaženje broja klastera za bilo koji skup uzoraka. Jednostavniji pristup je usporediti rezultate za veći broj primjena različitih *k* klasa i odabrati najpovoljniji rezultat. Na *Slika 5* dan je prikaz „pomicanja“ centroida u ovisnosti o blizini novim točkama.



Slika 5 - Formiranje centroida - sivo obojane točke su centroidi, a slovima označene točke su nove točke

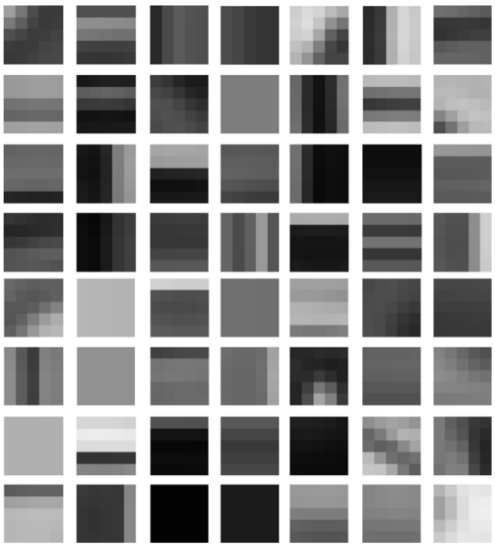
3.5 Metoda zbirki značajki (engl. Bag of words)

Metoda zbirki značajki (*engl. Bag of Words*) je metoda preuzeta iz jezične obrade (engl. *Natural LanguageProcessing*) u kojoj je ulazni parametar riječi (ili slova) u nekom tekstu, a izlazni parametar histogram tih riječi (bez veznika, priloga i sl) ili slova. Prikaz je dan na *Slika 6.*



Slika 6 - Histogram slova u riječima

Ako se pretpostavi da su značajke u slici „riječi“ slike tada bi se za određen niz slika mogao napraviti histogram sa frekvencijama pojave određene značajke. Osnovna ideja je da slične slike imaju slične značajke, kao i objekti na tim slikama te bi histogram značajki trebao biti sličan za određene slike. Budući da značajki može biti puno (kao i riječi), radi optimizacije algoritma potrebno je grupirati slične značajke [17].



Slika 7 - Prikaz nekih značajki slike

4. Značajke dokumenta

4.1 SIFT

SIFT (*engl. Scale Invariant Feature Transform*) temelji se na ekstrakciji značajki koje su invarijantne na rotaciju i mjerilo. Relativno je otporan na afine transformacije, varijacije u osvjetljenju i zaklonjenost objekata. SIFT posjeduje mogućnost detekcije karakterističnih značajki u slikama koje ostaju očuvane i nakon prethodno spomenutih transformacija. Dobivene značajke imaju poznatu lokaciju, mjerilo, orijentaciju i opisni vektor, te se mogu dalje koristiti za utvrđivanje korespondencije u slikama. Algoritam karakterizira mala vjerojatnost pogrešnog uparivanja korespondentnih značajki i uparivanje s velikim bazama podataka lokalnih značajki [9]. Osnovni postupak ovog algoritma može se podijeliti u nekoliko koraka. Na početku algoritma, koristeći razlike Gaussovim filtrom zaglađenih slika kao aproksimaciju Laplaceovog rubnog operatora, pronalaze se ekstremi invarijantni na mjerilo. Na taj način dobiva se veliki skup ekstrema iz kojeg je potrebno ukloniti one kandidate koji su nestabilni u pogledu niskog kontrasta, a osjetljivi su na šum i one kandidate koji su slabo lokalizirani uz rub. Nakon dobivanja skupa karakterističnih značajki slijedi dodjeljivanje jedne ili više orijentacija svakoj značajki iz skupa. Orijentacija se određuje računanjem gradijenata u okolini promatrane značajke, a osigurava invarijatnost na rotaciju. Na kraju algoritma svakoj značajki dodjeljuje se opisni vektor koji osigurava djelomičnu invarijatnost na varijacije u osvjetljenju, te promjene položaja kamere. Spomenuti postupak formira četiri osnovna koraka algoritma SIFT [9].

U prvom koraku određuju se slike u kojima de se tražiti potencijalni kandidati značajki. Na samom početku ulazna slika se zaglađuje Gaussovim filtrom sa standardnom devijacijom iznosa 0.5. Time se uklanjaju neželjeni učinci uslijed diskretizacije (engl. *aliasing*). Idući korak je povećanje rezolucije ulazne slike koristeći linearnu interpolaciju. D. Lowe predlaže da je dovoljno ulaznu sliku povećati samo jednom za faktor 2. Povećana slika se dalje zaglađuje Gaussovim filtrom jer ovaj filtar ne uvodi nove lažne strukture u krajnju sliku.

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{27}$$

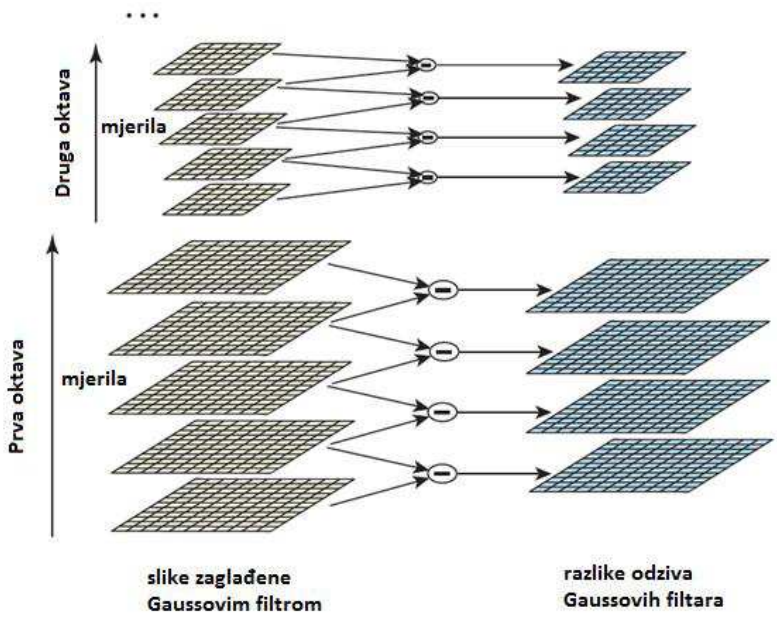
gdje je G Gauss-ov operator:

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2-y^2)/2a^2} \tag{28}$$

Kao što se vidi iz izraza (27) zaglađena slika L rezultat je konvolucije slike I i Gaussovog operatora G .

Što je veći σ u danom izrazu, to je veće zaglađivanje. U idućem koraku σ se poveća za faktor k, a kao rezultat dobije se zaglađena slika L za iduću razinu. U prostoru mjerila slike susjednih razina razlikuju se za faktor k. Sve slike jednake veličine po svim razinama mjerila čine oktavu. Nakon što se završi sa zaglađivanjem slike unutar prve oktave za sva mjerila, rezolucija slike se smanjuje za faktor 2. Smanjivanje slike izvodi se na način da se uzme svaki drugi slikovni element unutar retka i stupca slike. Smanjena slika predstavlja osnovnu sliku iduće oktave koja se dalje zaglađuje Gassovim filtrom za sva mjerila. Opisani se postupak ponavlja za svaku oktavu. Broj oktava i razina mjerila ovisi o veličini ulazne slike, a D. Lowe sugerira kako je 5 mjerila po 4 oktave dovoljno za uspješno detektiranje značajki [9].

Temeljem zaglađenih slika L koje čine Gaussovu piramidu (Slika 8 lijevo) generira se novi skup slika tzv. razlike Gaussiana (engl. *Difference of Gaussians, DoG*) prikazanih na Slika 8. desno.



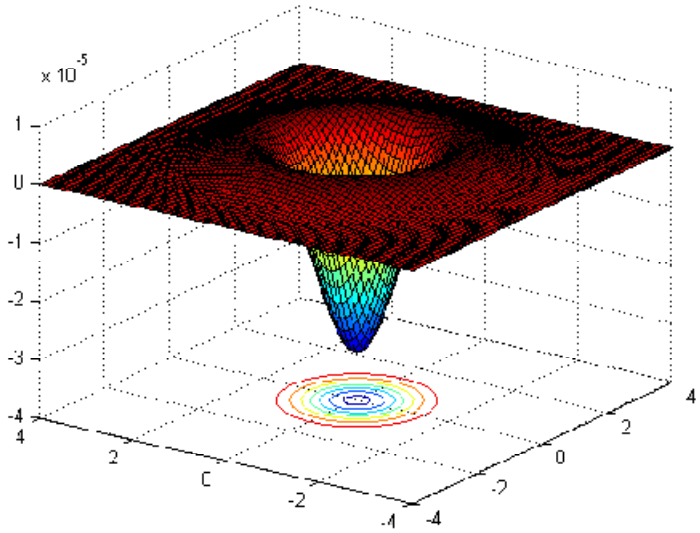
Slika 8 - Gauss-ova piramida

DoG slike D se računaju oduzimanjem susjednih slika unutar svake oktave Gaussove piramide. Njihov broj jednak je u svim oktavama, a za jedan manji od broja razina mjerila.

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \tag{29}$$

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma)) \tag{30}$$

Za otkivanje točaka maksimuma/minimuma u prostoru mjerila, koje su ujedno i najstabilniji kandidati značajki, nakon zaglađivanja Gaussovim filtrom primjenjuje se Laplaceov rubni operator (engl. *Laplacian of Gaussian, LoG*). Trodimenzionalni prikaz *LoG*-a dan je na Slika 9.



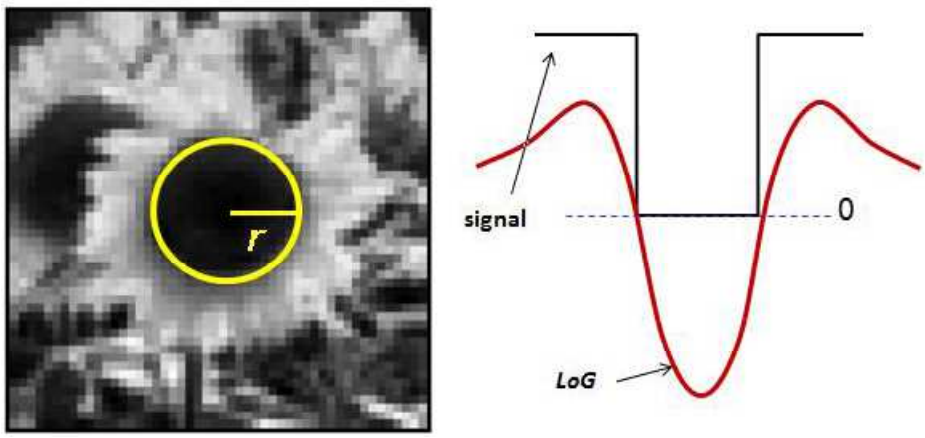
Slika 9 - Laplacian of Gaussian, LoG

LoG daje ekstremni odziv u regiji prilagođene veličine. Najveći odziv za krug prikazan na Slika 10, postiže se kad je odziv *LoG*-a poravnat sa krugom. *LoG* je definiran kao:

$$\nabla^2 G = \frac{\partial^2 G}{\partial^2 x^2} + \frac{\partial^2 G}{\partial^2 y^2} = (x^2 + y^2 - 2\sigma^2)e^{\frac{-(x^2+y^2)}{2a^2}} \tag{31}$$

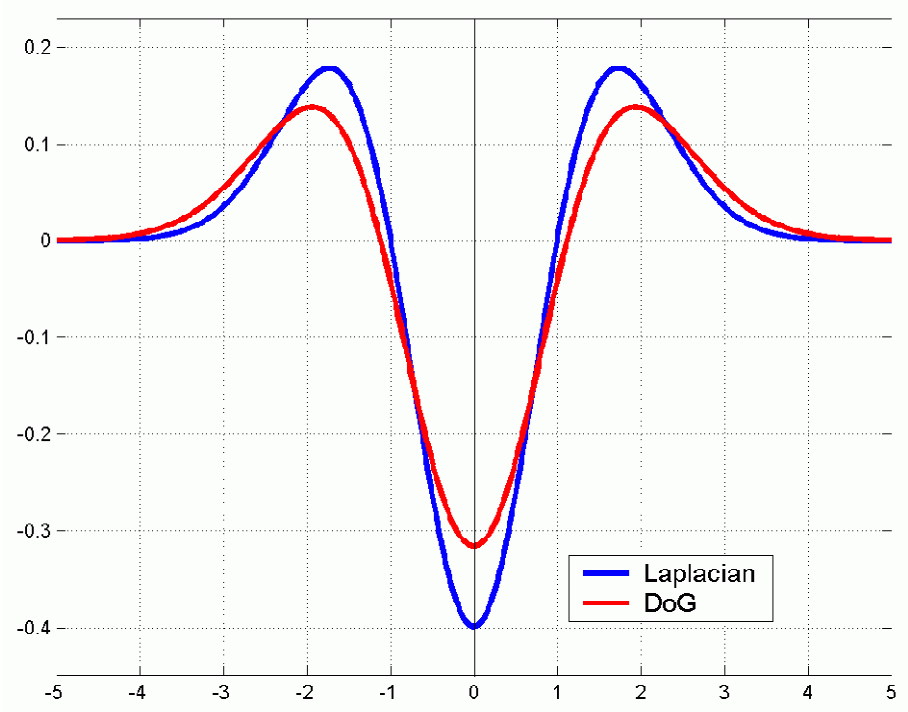
Zbog toga se maksimalan odziv *LoG*-a za binarni krug radijusa *r* postiže za mjerilo:

$$\sigma = \frac{r}{\sqrt{2}} \tag{32}$$



Slika 10 - Maksimum LoG-a

Iz izraza (31) vidi se da *LoG* zahtjeva proračun drugih parcijalnih derivacija što bi moglo nepogodno djelovati na brzinu cjelokupnog algoritma SIFT. Zbog toga se koriste *DoG* slike koje su dobra aproksimacija *LoG*-a. Na taj se način računanje parcijalnih derivacija svodi na jednu operaciju oduzimanja dviju zaglađenih slika.



Slika 11 - ***Aproksimacija LoG-a DoG-om***

Kako bi se pokazalo da je *DoG* zaista bliska aproksimacija *LoG* operatora (Slika 11) koji je normaliziran po mjerilu može se iskoristiti difuzna toplinska jednadžba koja je parametrizirana po σ :

$$\frac{\partial G}{\partial \sigma} = \sigma \nabla^2 \mathrm{G} \tag{33}$$

Pošto se $\frac{\partial G}{\partial \sigma}$ može aproksimirati pomoću diferencija, dobiva se slijedeća jednadžba:

$$\nabla^2 \mathrm{G} = \frac{\partial G}{\partial \sigma} \approx \frac{G(x, y, k\sigma) - G(x, y, \sigma)}{k\sigma - \sigma} \tag{34}$$

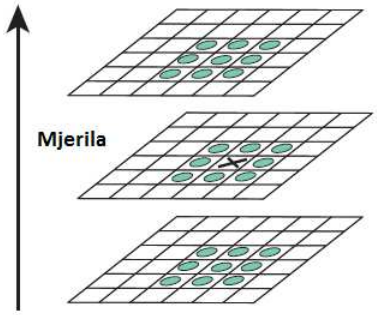
Iz čega slijedi:

$$(x, y, k\sigma) - G(x, y, \sigma) \approx (k-1)\sigma^2 \nabla^2 \mathrm{G} \tag{35}$$

Faktor $(k-1)$ u izrazu (35) konstantan je u svim mjerilima, te de stoga točke minimuma/maksimuma ostati na istoj lokaciji. Pogreška aproksimacije će idi u nulu kako se k približava jedinici. Istraživanje je pokazalo da čak i veće razlike između mjerila (npr. za k= 2) ne utječu na stabilnost detekcije ekstrema, kao ni na njihovu lokaciju [9].

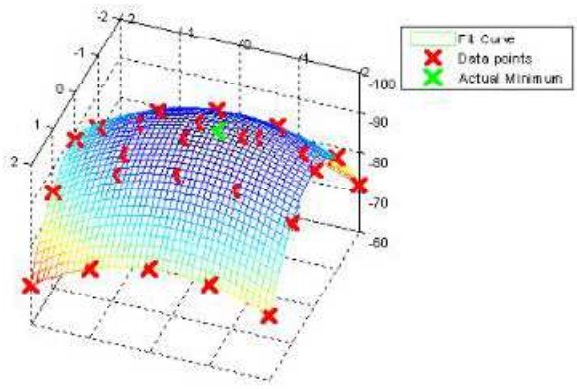
Detekcija i lokalizacija značajki

Da bi se kao krajnji rezultat ovog koraka dobile stabilne značajke, potrebno je prvo pronaći kandidate značajki. Kandidati se traže na temelju slika *DoG* iz prethodnog koraka.



*Slika 12 - **Traženje ekstrema u DoG***

Za sva mjerila po svim oktavama, izuzevši rubna mjerila, vrijednost svakog slikovnog elementa slike *DoG* uspoređuje se sa vrijednostima susjednih elemenata. Provjerava se da li je slikovni element slike *DoG* manji ili veći od svih elemenata u regiji 3x3 unutar trenutnog mjerila, te mjerila iznad i ispod (Slika 12). Ukoliko je pojedini slikovni element manji ili veći od svih 26 susjeda, tada on dobiva status kandidata značajke. Rezultat je velik broj kandidata koji su aproksimacija minimuma/maksimuma. Razlog je tome što se minimumi/maksimumi gotovo nikad ne nalaze točno na poziciji slikovnog elementa. Ekstremi se nalaze između slikovnih elemenata [19]. Na Slika 13 može se vidjeti jedan takav primjer. Crveni križići označavaju pozicije slikovnih elemenata, dok je pronađeni lokalni ekstrem označen zeleno.



Slika 13 - Minimum lociran između slikovnih elemenata

Međutim, kako se ne može pristupiti podacima između slikovnih elemenata, potrebno je matematički odrediti tu lokaciju. Lokacija se određuje Taylorovom ekspanzijom slike u okolini kandidata značajke [9].

$$\mathrm{D(x)} = \mathrm{D} + \frac{\partial^2 \mathrm{D^T}}{\partial \mathrm{x}}\mathrm{x} + \frac{1}{2}\mathrm{x^T}\frac{\partial^2 \mathrm{D}}{\partial \mathrm{x^2}}\mathrm{x} \tag{36}$$

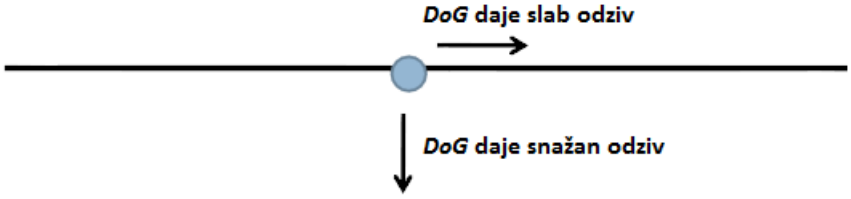
Gdje se D i njezine derivacije procjenjuju na temelju razlike susjeda kandidata značajke, a $x = (x, y, \sigma)^T$ je odmak od te značajke. Lokacija ekstrema x određuje se deriviranjem funkcije (36) obzirom na x i izjednačavanjem s nulom, što daje:

$$\hat{x} = -\frac{\partial^2 D^{-1}}{\partial x^2}\frac{\partial D}{\partial x} \tag{37}$$

Dobiva se lokacija značajke u smislu (x,y,σ), čime se povećava mogućnost podudarnosti i stabilnosti algoritma SIFT.
Ako je odmak x veći od 0.5 u bilo kojoj dimenziji, onda postoji neki drugi kandidat kojemu je promatrani ekstrem bliži. U tom slučaju lokacija ekstrema se mijenja i ponovno se radi interpolacija korištenjem Taylorovog reda drugog stupnja (vidi izraz 1.12). Konačna vrijednost odmaka x dodaje se lokaciji ekstrema kako bi se dobila interpolirana procjena lokacije kandidata značajke.
Ubacivanjem izraza (37) u (36) dobiva se:

$$D(\hat{x}) = D + \frac{1}{2}\frac{\partial D^T}{\partial x}\hat{x} \tag{38}$$

Vrijednost funkcije 38 koristi se za odbacivanje ekstrema niskog kontrasta. Odbacuju se svi ekstremi kojima je vrijednost $|D(\hat{x})|$ manja od 0.03. Na taj su način iz skupa kandidata značajki maknuti svi nestabilni kandidati podložni šumu. Kako bi se dobile samo stabilne značajke nije dovoljno odbacivanje ekstrema niskog kontrasta. *DoG* funkcija daje snažan odziv uz rub, a mali u ostalim smjerovima, kao što je prikazano na Slika 14.



Slika 14 - Odziv DoG-a

Postoji veliki broj kandidata značajki koji su i dalje nestabilni obzirom na male količine šuma. Za odbacivanje takvih kandidata iz preostalog skupa kandidata značajki koristi se Hessian matrica dimenzija 2x2.

$$\begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \tag{39}$$

Derivacije D_{xx}, D_{xy} i D_{yy} procjenjuju se na temelju razlike susjednih slikovnih elemenata kandidata značajke. Računanje svojstvenih vrijednosti matrice *H* može se izbjeći računanjem njihovog omjera. Ukoliko se svojstvene vrijednosti označe sa α i β, dobivaju se slijedeće jednadžbe (α je veća od njih):

$$Tr(H) = D_{xx} + D_{yy} = \alpha + \beta \tag{40}$$

$$Det(H) = D_{xx}D_{yy} - {D_{xy}}^2 = \alpha\beta \tag{41}$$

Ako je determinanta matice *H* negativna promatrani se kandidat odbacuje. U suprotnom se koristi usporedba omjera s određenim pragom r. Odnosno $\alpha{=}r\beta$, odakle slijedi:

$$\frac{Tr(H)^2}{Det(H)} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r} \tag{42}$$

Izraz (42) dokazuje kako nisu potrebne pojedinačne svojstvene vrijednosti, već samo njihov omjer. Omjer će biti najmanji u slučaju kad su svojstvene vrijednosti iste, a povećavat de se sa r. Na kraju preostaje provjeriti da li je izraz (42) istinit za $r{=}10$.

$$\frac{Tr(H)^2}{Det(H)} < \frac{(r+1)^2}{r} \tag{43}$$

U skupu ostaju samo stabilni kandidati i dobivaju status značajke. Svakoj značajki pridjeljuje se lokacija, odnosno x i y koordinate, oktava i mjerilo na kojem je pronađena, te vrijednost σ koja se računa prema formuli:

$$\sigma = \sigma_0 * 2^{o+\frac{s}{S}} \tag{44}$$

Indeks *o* označava oktavu, *s* mjerilo unutar oktave, a *S* broj intervala na koji je podijeljena svaka oktava. U pravilu *S*+3 predstavlja ukupan broj razina mjerila unutar oktave [23].

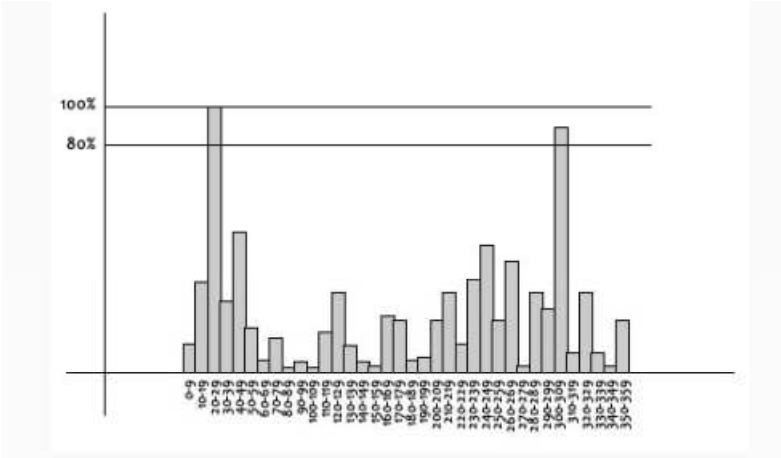
Dodjela orijentacije

U ovom koraku dodjeljuje se orijentacija svim značajkama koje su prošle selekciju iz prethodnog koraka. Orijentacija se dodjeljuje s obzirom na smjerove gradijenata tog dijela slike. Prvo se odabire Gauss-ovom funkcijom zaglađena slika L sa mjerilom koje odgovara promatranoj značajki. Na temelju vrijednosti susjednih slikovnih elemenata promatrane značajke, iz odgovarajuće slike L, dobiva se amplituda i kut gradijenta.

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \quad (45)$$

$$\theta(x,y) = tan^{-1}\left(\frac{L(x+1,y) - L(x-1,y)}{L(x,y+1) - L(x,y-1)}\right) \quad (46)$$

Amplituda, $m(x,y)$ i kut gradijenta, $\theta(x,y)$ računaju se prema izrazima (45) i (46) za svaki slikovni element u okolini značajke. Okolina, odnosno regija ovisi o mjerilu promatrane značajke, točnije o parametru $1.5*\sigma$. Zatim slijedi formiranje orijentacijskog histograma. Histogram se sastoji od 36 smjerova, pri čemu svaki smjer pokriva 10 stupnjeva, što daje ukupan raspon od 360 stupnjeva. Svaki izračun θ unutar regije značajke ponderiran vlastitim doprinosom dodaje se odgovarajućem smjeru u histogramu. Doprinos se računa na temelju Gauss-ove funkcije sa σ koja je 1.5 puta veda od mjerila promatrane značajke [23]. Nakon što je završeno računanje histograma, slijedi traženje smjera s najvećom vrijednosti u histogramu, tzv. vrh histograma. Vrh s pridruženim smjerom odgovara dominantnoj orijentaciji u histogramu. No, sam vrh nije dovoljan da bi se odredila orijentacija promatrane značajke. Zato se u izračun orijentacije uzimaju i svi lokalni vrhovi, odnosno smjerovi čije su vrijednosti unutar 80% najveće vrijednosti (Slika 15). Ukoliko postoji više takvih vrhova podjednake amplitude, promatranoj značajki pridjeljuje se više orijentacija. Stvaraju se nove značajke alternativnog smjera sa istom lokacijom i mjerilom. Prema istraživanju, to se događa samo u 15% slučajeva i značajno pridonosi stabilnosti podudaranja [23], [24].



Slika 15 - Histogram orijentacija

Na kraju se konstruira parabola kroz tri točke histograma. Za središnju točku uzima se vrh histograma, dok ostale dvije odgovaraju najbližim susjedima lijevo i desno od vrha. Na taj način vrši se kvadratna interpolacija, te se dobiva finija raspodjela orijentacije.

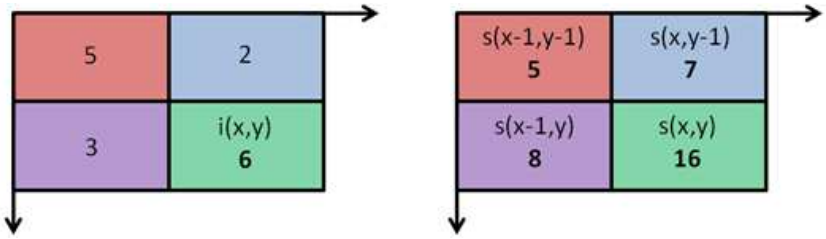
4.2 SURF detektor

SURF detektor (*engl.Speeded Up Robust Feature* - ubrzani robusni detektor značajki) je detektor značajki koji je djelomično izveden iz SIFT detektora, ali za razliku od SIFT-a, nekoliko je puta brži [18]. Kasnije u radu pokazane su neke od razlika SURF-a nad SIFT-om sa kojima su dobivene pozitivne razlike u performansi. Također, autori SURF-a navode da je njihov detektor nekoliko puta efikasniji prilikom detekcija značajki transformiranih slika (rotacija, translacija, faktor proporcionalnosti itd). SURF se temelji na sumama odaziva dvodimenzionalnih Haar wavelet-a te za detekciju koristi integralne slike.

Jedan veliki dio poboljšanja performansi u SURF deskriptoru može se pripisati korištenju integralnih slika. Svaki piksel na slici ima određeni intenzitet koji se može numerički predstaviti. Na taj način digitalnu sliku možemo predstaviti kao tablicu, gdje svaka ćelija ima određeni intenzitet. Kod integralnih slika, slika se također predstavlja tablicom, ali je u svakoj ćeliji zapisana vrijednost sume intenziteta iznad ćelije, lijevo od ćelije te intenziteta ćelije koju se promatra. Formalni zapis je sljedeći:

$$I_{\Sigma}(x,y) = I(x,y) + \sum_{i=0}^{i\leq x} I(x,y) + \sum_{j=0}^{j\leq y} I(x,y) - \sum_{i=0}^{i<x}\sum_{j=0}^{j<y} I(x,y) \tag{47}$$

Korištenjem integralnih slika, računanje područja pravokutnog oblika, svedeno je na četiri operacije. Jednostavan primjer predstavljanja originalne slike integralnom slikom prikazan je na Slika 16 [19].

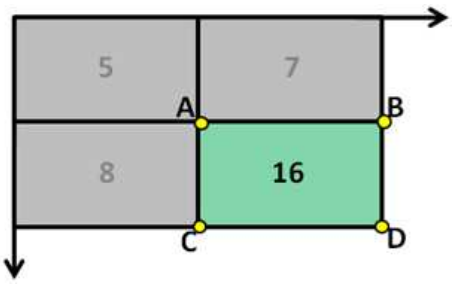


Slika 16 - Integralni prikaz slike

Isto tako, u svakom trenutku moguće je iz integralne slike izračunati vrijednost intenziteta promatranog piksela originalne slike:

$$I(x,y) = I_{\Sigma}(x,y) - \sum_{i=0}^{i\leq x} I(x,y) - \sum_{j=0}^{j\leq y} I(x,y) + \sum_{i=0}^{i<x}\sum_{j=0}^{j<y} I(x,y) \tag{48}$$

Ako se rubne točke ćelija označe slovima A, B, C i D moguće je dobiti prikaz kao na Slika 17



Slika 17 - Izračun vrijednosti pojedine ćelije integralne slike

Gdje je područje A ćelija lijevo i iznad od promatrane ćelije *(Slika 17*), B ćelija iznad (7), C ćelija lijevo (8), a D vrijednost intenziteta promatrane ćelije (16).
Tada izraz (48) prelazi u:

$$I(x,y) = A + D - B - C \tag{49}$$

SURF detektor temelji se na vrijednosti determinante Hessian matrice u pojedinoj točki. Definicija Hessian matrica dana je sljedećim izrazom [20]:

$$H\big(f(x,y)\big) = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix} \tag{50}$$

Determinanta takve matrica dana je sljedećim izrazom:

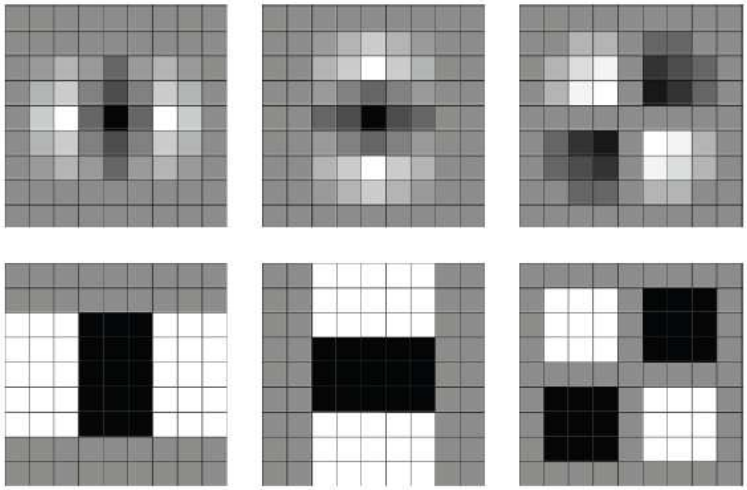
$$\det(H) = \frac{\partial^2 f}{\partial x^2}\frac{\partial^2 f}{\partial y^2} - \left(\frac{\partial^2 f}{\partial x \partial y}\right)^2 \tag{51}$$

Budući da se u obradi slike radi sa diskretnim vrijednostima, a ne kontinuiranim funkcijama potrebno je Hessian matricu (tj njene članove) prikazati u diskretnom obliku.
Svaka točka na slici (piksel) može se opisati razinom osvjetljenja (intenzitetom) te se funkcija *f(x,y)* može zamijeniti funkcijom *I(x,y).* Parcijalnu derivaciju drugog reda moguće je zamijeniti aproksimacijom tj. Gaussian-om drugog reda. Ovu zamjenu može se promatrati kao filter (kernel) funkciju. Točnije, radi se konvolucija svakog piksela (i njegovog okruženja) sa kernel funkcijom Gaussian-a. Izraz (50) je sad moguće zapisati na sljedeći način:

$$H(\mathrm{x},\sigma) = \begin{bmatrix} L_{xx}(\mathrm{x},\sigma) & L_{xy}(\mathrm{x},\sigma) \\ L_{xy}(\mathrm{x},\sigma) & L_{yy}(\mathrm{x},\sigma) \end{bmatrix} \tag{52}$$

Gdje seL_{xx}odnosi na konvoluciju Gaussian derivacije drugog reda ($\frac{\partial^2 g(\sigma)}{\partial x^2}$) sa slikom u točki *x = (x,y)*. Vrijednost *σ* se odnosi na veličinu filtera. Slično je i za ostale članove Hessian matrice zapisane na ovaj način. Ovako zapisane parcijalne derivacije poznate su kao Laplaciani Gaussiana (LoG).
Kod SURF algoritma filteri korišteni za konvoluciju su pojednostavljeni u odnosu na pravi LoG te je time dobiveno na robusnosti (nazivaju se aproksimacije LoG-a) [20].

Na *Slika 18*prikazani su filteri pravog LoG-a i pripadnih aproksimacija koji SURF koristi.

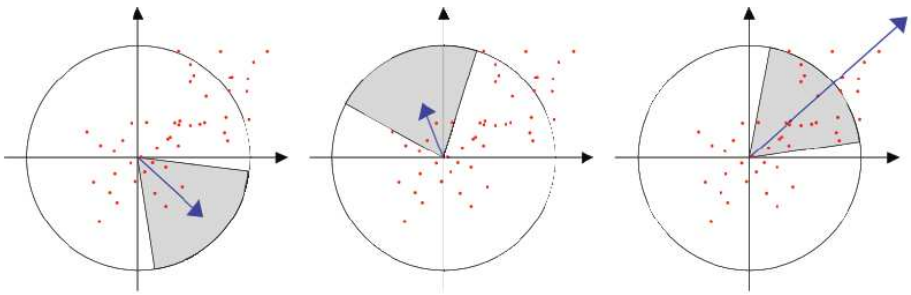


Slika 18 - Filteri LoG-a

Na *Slika 18* u gornjem redu prikazani su sa lijeva na desno LoG filteri u smjeru *x* (L_{xx}), *y*(L_{yy}) i *x-y*(L_{xy}). U redu ispod, prikazane su aproksimacije gornjih filtera u pripadnim smjerovima označenimD_{xx}, D_{yy}, D_{xy}. U smjeru D_{xx} i D_{yy}vrijednost bijelog područja jest 1, a crnog područja 2, dok za smjer D_{xy}bijelo područje ima vrijednost 1, a crno područje -1.

Ovisno o vrijednosti σ određuje se područje oko interesne točke. Budući da je SURF metoda invarijabilna na transformacije rotacije i proporcionalnost, također je bitan i smjer u kojem se orijentira područje značajke.

Nakon što se značajke pronađu dobiveni su podaci o lokaciji i veličini značajke. Kako bi se očuvala invarijatnost metode u odnosu na rotaciju potrebno je odrediti i smjer značajke. Smjer se određuje korištenjem Haar wavelet-a u području oko značajke (točnije filteri veličine 4σ u području radijusa 6σsa središtem u točki značajke). Odazivi Haar wavelet-a predstavljeni su svojim težinama Gaussiana. Nakon dobivenih odaziva, u području kruga kreira se „prozor“ veličine π/3 sa kojim se kruži oko središta te se smjer dodjeljuje tamo gdje se nalazi najviše točaka odaziva. Primjer je dan na *Slika 19*.



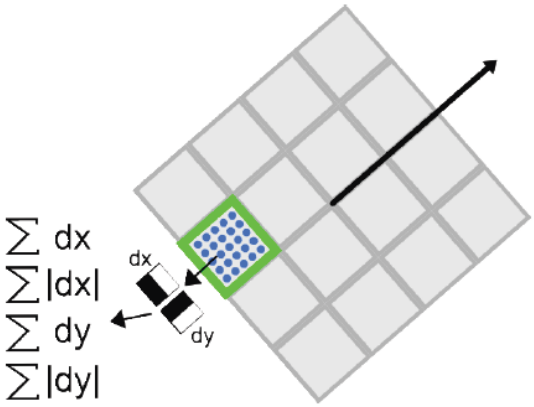
Slika 19 - Određivanje orijentacije značajke

Nakon što algoritam pronađe značajke i pripadnu orijentaciju, potrebno je iste ne neki način opisati. Na taj način se postiže univerzalnost značajke te je kasnija obrada značajki pojednostavljena.

Prvo je potrebno formirati kvadrat oko značajke sa značajkom u samoj sredini. Veličina kvadrata ovisio parametru σ („veličina“ značajke). U takvom kvadratu sadržane su točke koje će opisivati značajke. Ulazni parametar kod kreiranja deskriptora upravo su sve točke unutar kvadrata koji je veličine 20σ. Kvadrat se postavlja na način da je njegova orijentacija u smjeru definirane orijentacije značajke. Kvadrat je podijeljena na 16 manjih kvadrata (4 x 4). Unutar svakog manjeg kvadrata računaju se odazivi Haar wavelet-a veličine 2σna 25 jednako distribuiranih točaka unutar tog manjeg kvadrata. Odaziv Haar wavelet-a u *x* i *y* smjeru se označava *dx* i *dy*, a ukupno se dohvaća 4 podatka:

$$v_{sub} = \left[\sum dx, \sum dy, \sum |dx|, \sum |dy|\right] \tag{53}$$

Dakle za svaki manji kvadrat postoje četiri veličine te takvih manjih kvadrata ima ukupno 16 unutar jednog velikog kvadrata. Prema tome, za svaku značajku postoje 64 podatka koja se spremaju u polje. Prikaz je dan *Slika 20.*



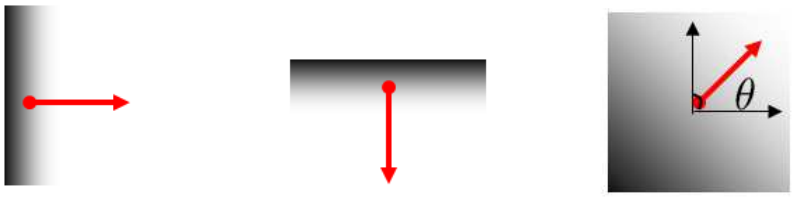
Slika 20 - Deskriptor značajki

4.3 Histogram orijentiranih gradijenata (HOG)

Gradijent u slici opisuje promjenu intenziteta na određenom području. Najčešće se za svaku točku slike gleda susjedstvo te točke te promjene intenziteta u horizontalnom i vertikalnom smjeru. Definicija gradijenta dana je izrazom (54):

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right] \tag{54}$$

Prikaz gradijenta dan je na *Slika 21.*



$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$ a) $\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$ b) $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$ c)

Slika 21 - Prikaz gradijenta u smjeru x (a), smjeru y (b) i oba smjera (c)

Smjer gradijenta moguće je odrediti izrazom (2).

$$\theta = arctg\left(\frac{\partial f/\partial x}{\partial f/\partial y}\right) \tag{55}$$

Osim smjera gradijenta moguće je odrediti i amplitudu gradijenta. Isti je dan izrazom (56).

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \tag{56}$$

Gradijent se najčešće koristi na području računalnog vida prilikom određivanja rubova na slici. Smjer gradijenta definira smjer okomit na rub, a amplituda smjera govori o „jačini“ ruba. Izrazi (54), (55) i (56) su definicija gradijenta na slici koja je opisana kontinuiranom varijablom.

Za rad sa digitalnim slikama potrebno je navedene izraze transformirati u diskretno područje. U diskretnom području promjena intenziteta piksela (derivacija) može se izračunati kombiniranjem slike u digitalnom obliku i nekog od derivacijskih filtera (na primjer Sobel filter). U svakoj točki slike izračuna se konvolucija sa željenim filterom, a rezultantna slika je prikaz svih gradijenata.

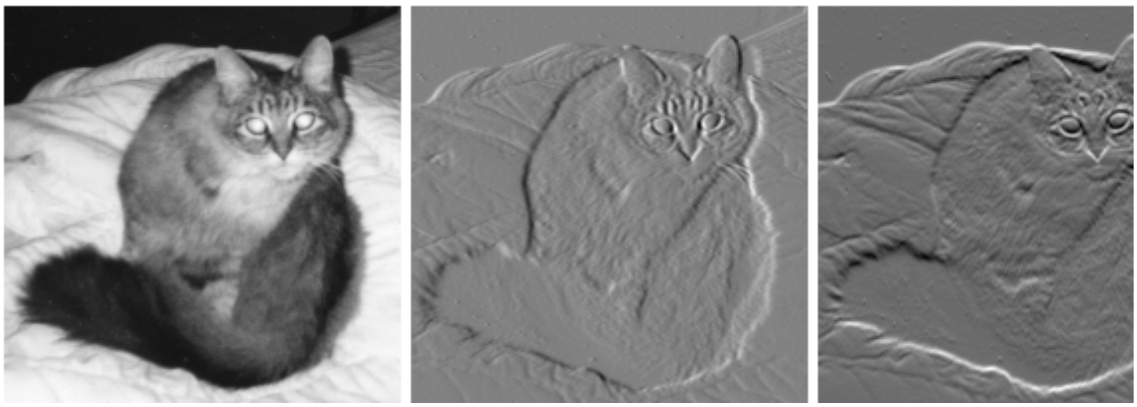
Konvolucija je nad kontinuiranim funkcijama definirana izrazom (57).

$$h(t) = f * g = \int_{-\infty}^{\infty} f(\tau)g((t-\tau)d\tau \tag{57}$$

Kao što je vidljivo iz izraza (4), rezultat konvolucije dvije funkcije je nova funkcija. U diskretnom području konvolucija je definirana izrazom (58).

$$h(t) = \sum_{\tau} f(\tau)g((t-\tau) \tag{58}$$

U ovom radu za konvolucijski filter određivanja gradijenata korišten je jednostavni filter u horizontalnom smjeru $[-1\ 0\ 1]$ i vertikalnom smjeru $[-1\ 0\ 1]^T$. Primjer slike obrađene ovim filterima dan je na *Slika 22.*



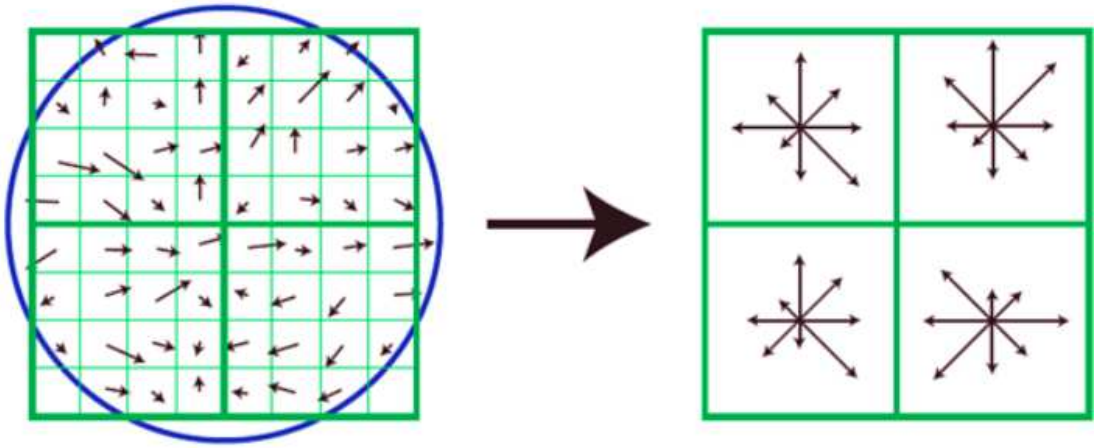
a) *b)* *c)*

Slika 22 - Primjer obrade slike (2a) sa filterima $[-1\ 0\ 1]$ $(2b)$ *i* $[-1\ 0\ 1]^T$ *(2c)*
(Izvor: http://en.wikipedia.org/wiki/File:Intensity_image_with_gradient_images.png)

Histogram orijentiranih gradijenata računa se kako je opisano u [21]. Osnovna ideja je podjela slike na manje dijelove (blokove) koji se sastoje od ćelija koji se opet sastoje od piksela. Za svaku ćeliju se računa histogram orijentiranih gradijenata. Postupak je detaljnije opisan u nastavku.

Blokovi i ćelije

Ćelija je definirana kao površina od *(n* x *n)* piksela. Blok je definiran kao površina od *(m x m*) ćelija. Gradijent se računa za svaki piskel u ćeliji, a za svaku ćeliju se računa histogram orijentiranih gradijenata. Prikaz određivanja histograma dan je na *Slika 23*.



Slika 23 - Određivanje gradijenata u bloku veličine 2 x 2 ćelije, ćelije veličine 4 x 4 piksela
(Izvor: http://farshbafdoustar.blogspot.com/2011/09/hog-with-matlab-implementation.html)

U ovom radu blokovi i ćelije su definirane na sljedeći način:

- Ćelija : 8 x 8 piksela
- Blok: 16 x 16 piksela (2 x 2 ćelije)

Prema [21], histogram se sastoji od 9 stupaca širine 20 stupnjeva. Prilikom računanja vrijednosti pojedinog stupaca, vektorima gradijenta koji su negativnog predznaka (od 0 do -180 stupnjeva) pridjeljuje se pozitivni predznak te se takvi vektori dodaju onima sa područja od 0 do 180 stupnjeva. Ovime je pojednostavljen zapis histograma, a podaci o gradijentu su sačuvani. Visina pojedinog stupca ovisi o smjeru i amplitudi gradijenta, sa tim da je amplituda težinski faktor (veću zastupljenost u histogramu će imati smjer sa većom amplitudom).

Normalizacija i obrada bloka

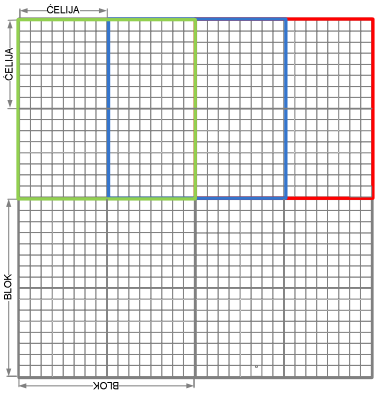
Prilikom računanja histograma, pojedini blokovi se lokalno normaliziraju. Normalizacija se izvodi kako bi se uračunale promjene svjetline i kontrasta. Prema [21], optimalna normalizacija se pokazala L2-normalizacija.
Ako je v neki vektor koji sadrži histograme bloka, a nije normaliziran, neka je $\|v_k\|$ $k-$ norma za $k = 1{,}2$ i neka je e neka zanemarivo mala konstantna vrijednost. Faktor normalizacije se tada može izračunati kao:

$$L2 - norma = f = \frac{v}{\sqrt{\|v\|_2^2 + e^2}} \tag{59}$$

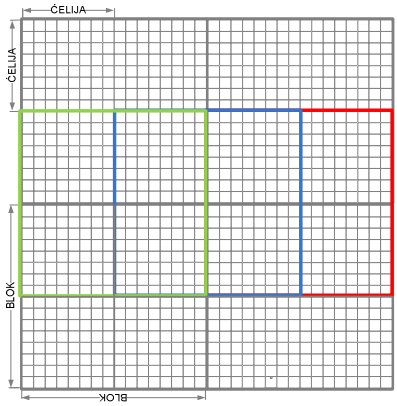
Također, prema [21], manja poboljšanja performansa dobivena su primjenom Gaussovog filtera nad svakim blokom, prije samog određivanja gradijenta.

Preklapanje blokova

Kako bi pojedini pikseli imali (tj područje koje opisuju) podjednaku zastupljenost krajnjem deskriptoru (histogramu svih blokova), autori [22] uvode i preklapanje blokova. U ovom radu preklapajući blok je veličine 16 x 16 piksela (isti kao i osnovni blok).
Ako se uzme slika veličine 32 x 32 piksela, blok 16 x 16 piksela (2 x 2 ćelije), ćelija 8 x 8 piksela, preklapanje 16 x 16 piksela, deskriptor će imati ukupno 9 blokova, po 4 histograma sa 9 stupaca (Slika 24).



a)

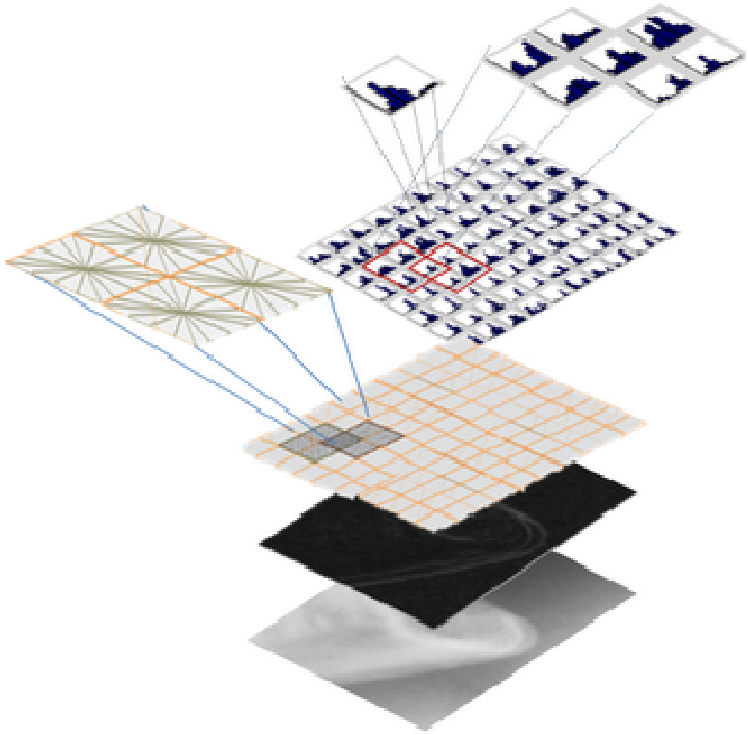


b)

c)

Slika 24 - 9 blokova za sliku dimenzije 32 x 32 - a) 3 bloka u gornjem redu, b) 3 u srednjem te c) 3 u donjem redu

Dakle, deskriptor za ovakvu sliku bi imao ukupno 324 elemenata (9 stupaca x 4 ćelije x 9 blokova), gdje svaki element predstavlja stupac histograma. Poredak dohvata blokova je sa lijeva na desno i od gore prema dolje. Ako se želi dohvatiti histogram prve ćelije, potrebno je dohvatiti prvih 9 elemenata deskriptora.
Opisna slika cijelog procesa računanja histograma orijentiranih gradijenata dana je na *Slika 25.*



Slika 25 - Prikaz procesa nalaženja deskriptora HOG-a
(izvor: http://farshbafdoustar.blogspot.com/2011/09/hog-with-matlab-implementation.html)

4.4 Pečati kao značajka dokumenta

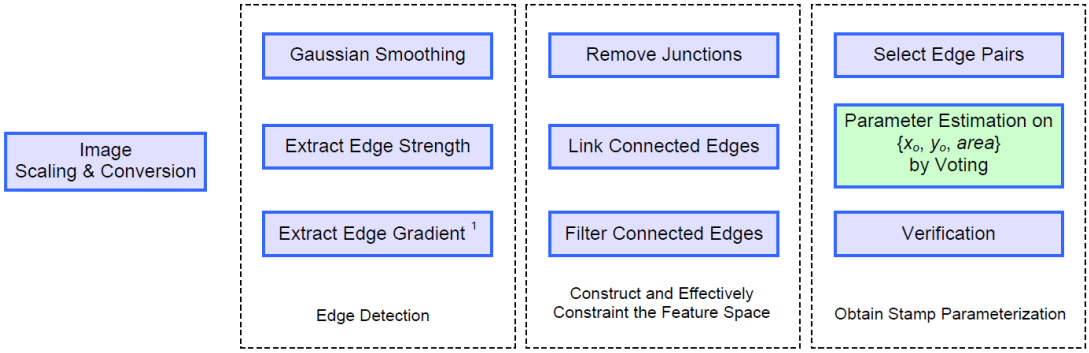
Pečati u dokumentima predstavljaju autentičnost dokumenata te istraživanje pečata na dokumentima ima veliku primjenu, od verifikacije identiteta i

autentičnosti, preko otkrivanja falsifikata pa do klasifikacije. Budući da pečat i potpis sadrže podatke o vlasniku ili autoru dokumenta, njihovo istraživanje u obradi slike dokumenata se nameće samo po sebi.

Detekcija pečata na slikama je problem koji je istraživan u više radova [27][28][29]. Jedan od osnovnih problem prilikom detekcije pečata je definicija problema. Pečati sadrže jedinstven niz karakteristika koji otežavaju primjenu klasičnih detektora oblika (na primjer logoa u dokumentima). Pečati su općenito nestabilni u svom geometrijskom kontekstu. Čak uz strogu definiciju izgleda službenih (na primjer državnih) pečata, slika pečata na realnim dokumentima može biti znatno drukčija (kut pečata, pritisak, degradacija boje i sl). Nadalje pečati imaju puno manju prostornu gustoću od logoa ili ostalih uzoraka u slikama dokumenta. Čak iz uz očuvane zatvorene konture pečata, isti se pojavljuju kao slabije regije u segmentaciji slike u odnosu na tekst ili slike.

Također, slika pečata može se nalaziti bilo gdje na dokumentu, od idealnih prostora bez šumova, do prostora sa izrazitim šumom koji otežava detekciju.

Autori [28] predlažu rješenje za detekciju degradiranih pečata na starijim (degradiranim) dokumentima. Rješenje se sastoji od tri faze prikazane na Slika 26.



Slika 26 - Detekcija degradiranih pečata na dokumentima [28]

Prije same obrade, slika dokumenta se smanjuje te se rezolucija smanjuje sa ciljem smanjivanje količine podataka koji ulaze u algoritam. Odabir omjera smanjivanja slike je izuzetno bitan radi očuvanja rubova koji se kasnije koriste u analizi slike.

Prva faza nakon smanjivanja slike je Gaussian izglađivanje i detekcija rubova te dobivanje njihovih međusobnih težinskih vrijednosti i gradijenata.

U drugoj fazi uklanjaju se sjecišta rubova sa ciljem izbjegavanja "dvosmislenih" rubova. Recimo, ako se pečat nalazi na području gdje je i potpis, navedenim postupkom pokušava se prekinuti zatvorena kontura slike pečata i potpisa. Zatim se istražuju rubovi koji spadaju u istu konturu (spojeni rubovi). Filtriraju se samo rubovi koji se smatraju spojenim.

U trećoj fazi autori predlažu detektor eliptičnih oblika. Standardni algoritmi za detekciju eliptičnih oblika mogu se podijeli u dvije grupe. U prvoj grupi je Hough-ova

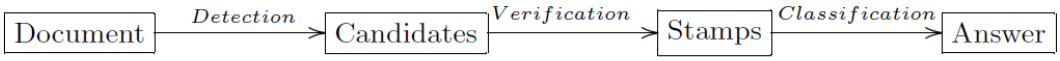
transformacija i varijacije istog [29], [30], [31], [32], zatim RANSAC [33] i neizrazita logika (engl. *fuzzy logic*) [34], [35]. Navedeni su detektori robusni po pitanju okluzije, ali su performansno skupi zbog parametrizacije sa 5 članova. Druga grupa uključuje least square fitting algoritam [35] i genetske algoritme [36] koji detektiraju eliptične oblike svojstvenim funkcijama, ali uključuju prethodno procesiranje poput segmentacije ili grupiranja.
Detektor koji predlažu autori [28] u je zapravo detektor središta eliptičnog oblika. Navedenim pristupom, traženjem centra potencijalnog eliptičnog oblika između dva spojena ruba, postiže se linearan izračun (za razliku od standardnog Hough algoritma, koji ima polinom drugog stupnja).

Još jedan pristup u detekciji pečat dali su i autori [38]. Njihov pristup je drugačiji u odnosu na autore [28] u tome što se detekcija vrši nad pečatima svih oblika, dok je ranije spomenuti detektor otkriva samo ovalne pečate. Također, ističu da ne koriste Hogh-ovu transformaciju niti varijante istih za detekciju eliptičnih oblika.
Autori dijele tipove pečata u dvije klase: službeni i neslužbeni pečati. Službeni pečati su najčešće jednostavnijih oblika (krug, trokut, četverokut) dok su neslužbeni pečati složenijih oblika (Slika 27 i Slika 28).

Slika 27 - Službeni pečati [38]

Slika 28 - Neslužbeni pečati [38]

Nadalje, definiraju dvije grupe značajki koje opisuju pečate. Prva grupa je ona sa prostornim značajka , uključujući dimenzije (proporcije), distribuciju rubova, srednje vrijednosti i varijance gradijenta, momente. Druga grupa uključuje svojstva boja, poput distribucije boja HSV i YCbCr prostora.
Predloženi algoritam se sastoji od nekoliko dijelova: detekcija, verifikacija i klasifikacija (Slika 29).



Slika 29 - Koraci algoritma [38]

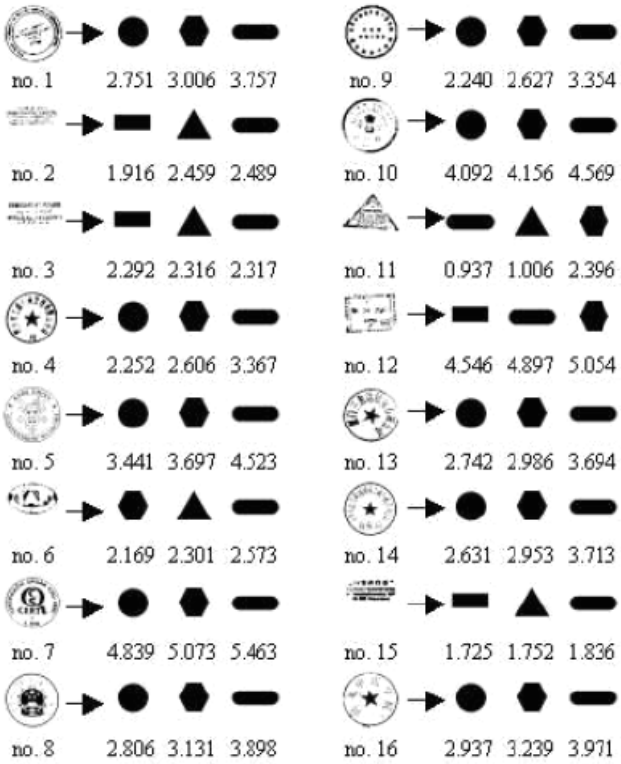
U fazi detekcije, slika se dijeli na Cr i Cb komponente YCrCb prezentacije. Autori polaze od pretpostavke da su službeni pečati najčešće crvene ili plave boje (Slika 30).

Slika 30 - Komponente slike dokumenta[38]

Vidljivo iz Slika 30, ekstremi u svjetlini su u području pečata. Tako obrađena slika predaje se sljedećoj fazi verifikacije gdje se provjeravaju kandidati. Vrše se projekcije slike u x i y smjeru sa ciljem dohvaćanja ekstrema.

Na područjima ekstrema se promatraju dimenzije ekstrema. Površina na kojoj se nalazi ekstrem ne smije biti manja od 5% ili veća od 15% površine dokumenta. Samo takva područja se smatraju kandidatima.

U posljednjoj fazi vrši se detekcija oblika pečata korištenjem algoritma općenite analize oblika (engl. *General shape detection*) [39]. Algoritam u svojoj osnovi promatra svojstva predanog mu objekta u smislu dimenzija, rasporeda rubova te ga uspoređuje sa nekim od osnovnih geometrijskih oblika (na primjer krug, pravokutnik, elipsa ili trokut). Pomoću sustava glasovanja, promatranom objektu se dodjeljuju više vrijednosti gdje najmanja vrijednost označava najveću vjerojatnost pripadanja promatranog oblika nekom od osnovnih oblika (Slika 31).

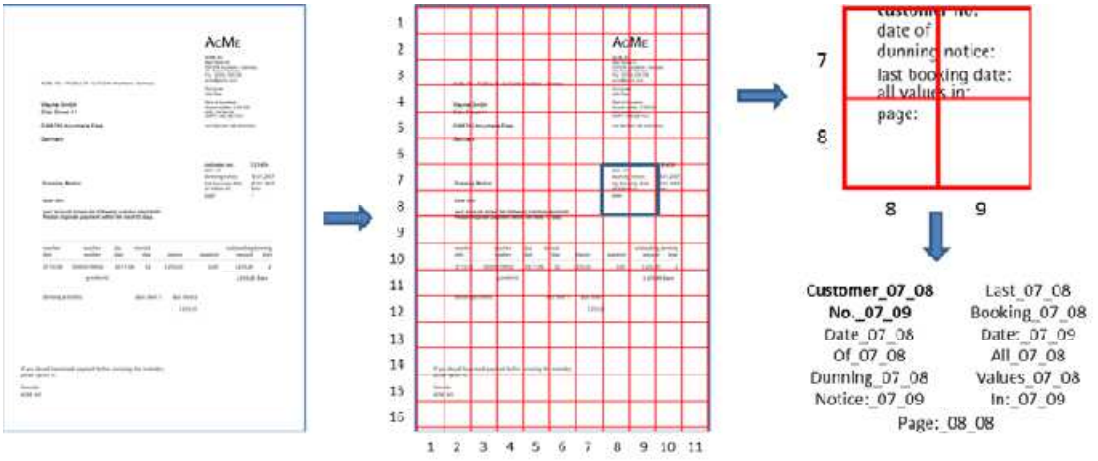


Slika 31 - Dodijeljivanje općih oblika pečatu[39]

4.5 Klasifikacija dokumenta korištenjem svojstava pečata

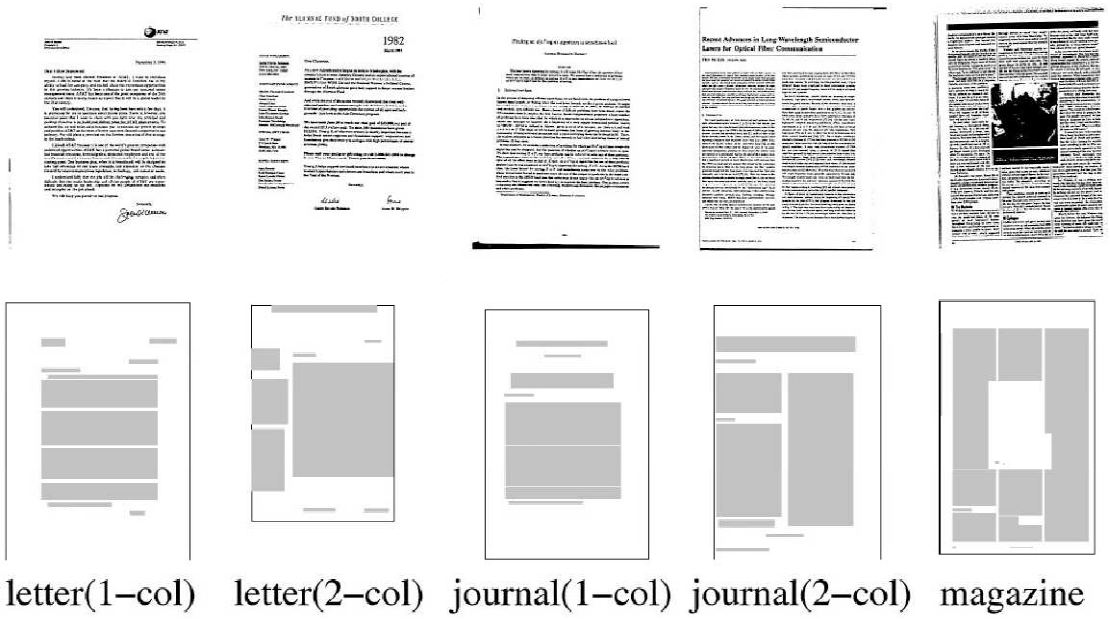
Klasifikacija dokumenata u smislu obrade slike, danas se većinom odvija korištenjem nekih o OCR alata. U pravilu, očitava se cijeli dokument i nad istim se vrši OCR očitavanje te se koristeći nekih od algoritama za dubinsko istraživanje podataka (engl. Data Mining), riječi i tekst nadalje se obrađuju u svrhu klasifikacije.

Jedan od pristupa sa OCR metodologijom koji djelomično koristi prednosti strojnog učenja jest istraživanje predložaka dokumenata i učenje o predlošcima koje se koristi kao polazišna točka za klasifikaciju dokumenta. Jedan od takvih pristupa je i opisan u [40] gdje autori koriste predefinirane predloške te algoritam za učenje treniraju za prepoznavanje predložaka. Na temelju klasifikacije predloška, određene regije se obrađuju OCR metodom te se dohvaća tekst iz određene regije (Slika 32).



Slika 32 - Dodjeljivanje indeksa regijama dokumenta[40]

Još jedan sličan pristup usporedbe predložaka dokumenata dali su autori [41]. Predloženo rješenje određenom dokumentu dodjeljuje određen faktor sličnosti nekom ranijem definiranom predlošku (Slika 33).

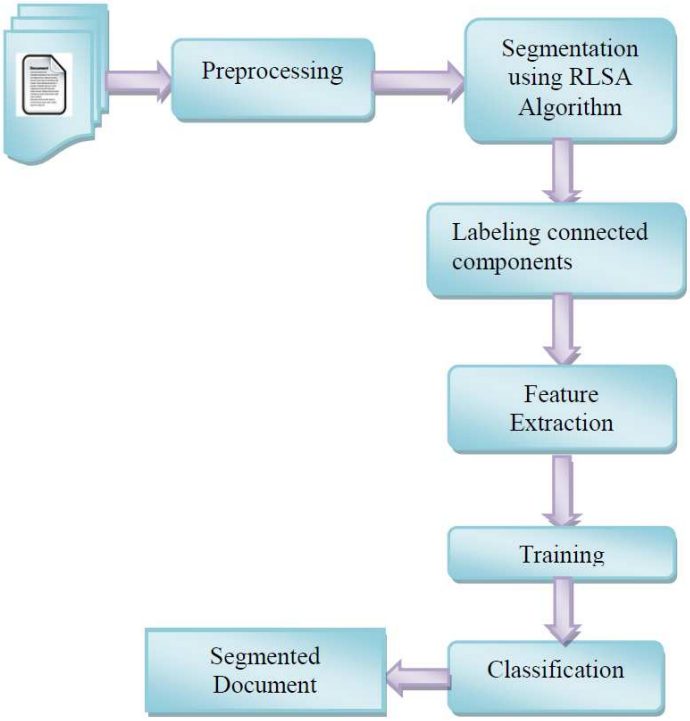


Slika 33 - Tipovi predložaka[41].

U navedenom rješenju prednosti su u tome što se ne koristi OCR metoda prilikom obrade slike već samo položaju pojedinih elemenata na dokumentu (na primjer zaglavlje, tijelo, stupci i slično). Ovakav algoritam se može nadalje nadograditi klasifikatorom koji bi dobivene predloške i regije istih pretraživao po određenim značajkama. Autori također navode i relativno brzu obradu predloška jer rezolucija slike nije nužno visoka što smanjuje količinu podataka na ulazu u algoritmu.

Sličan pristup gore navedenom (traženje klasa predložaka i regija) imaju i autori [42] koji koriste metodu višeslojnih perceptrona (engl. Multilayer Perceptron - MLP) za segmentaciju i pronalaženje regija dokumenata (Slika 34). Njihov se algoritam sastoji

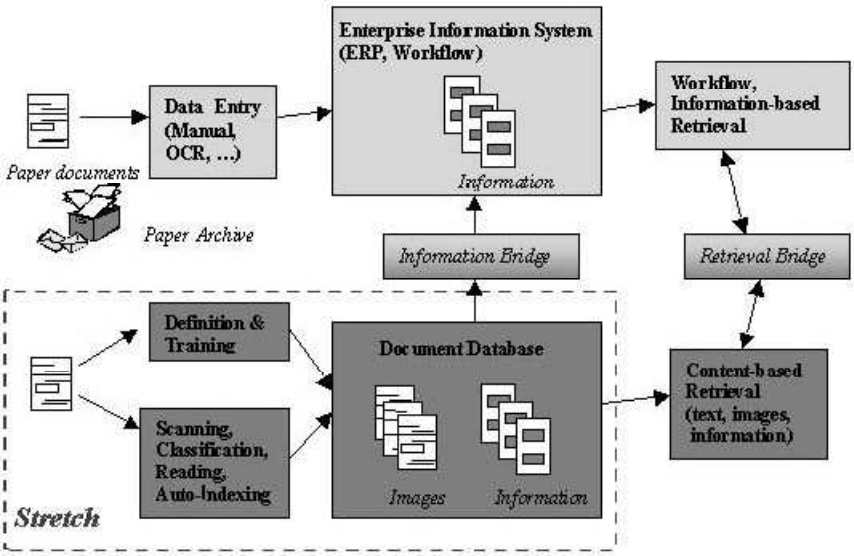
od više koraka u kojem je ključni primjena MLP-a algoritma (proizašlog iz teorije neuronskih mreža) te izlučivanje značajki iz povezanih blokova slike.



Slika 34 - Koraci algoritma[42]

Slike se prvo pretvaraju u binarne zatim se primjenjuje RLSA algoritam za segmentaciju (engl. Run length smearing algorithm). Nakon segmentacije komponente se povezuju i označavaju. Zatim se iz dobivenih spojenih komponenti izvode značajke koje ulaze u trening fazu MLP algoritma. Nakon toga vrši se klasifikacija dokumenta.

Postoje i rješenja za klasifikaciju dokumenata u velikim sustavima, koja su najčešće komercijalna. Kao takva, ne opisuju primijenjene algoritme u obradi i klasifikaciji dokumenta već samo proces. Jedno od takvih rješenja je implementirano u [43] pod nazivom STRECH (engl. STorage and RETrieval by Content of imaged documents). Na Slika 35 prikazana je shema sustava.



Slika 35 - STRECH sustav[43]

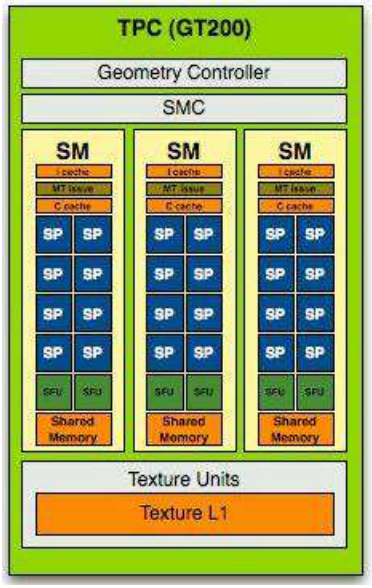
Navedeno rješenje prilikom klasifikacije koristi stabla odlučivanja, a prednost sustava se nalazi u tome što za ulaz prima više oblika digitalnih dokumenta: od samog teksta (recimo dobivenog OCR metodom ili ručno unesenim) do slike dokumenta. Sustav se također ne bazira na OCR pristupu prilikom istraživanja predloška nego koristi razrađeni sustav stabla odlučivanja kod definicije pojedine regije u trening fazi.

5. GPU arhitektura

Prije nekoliko godina tvrtka NVIDIA predstavlja novu arhitekturu za paralelno programiranje pod imenom CUDA (engl. *Compute Unified Device Architecture*). Radi se o paralelnoj arhitekturi s novim paralelnim programskim modelom i skupom instrukcija, koja daleko pojednostavljuje paralelno programiranje na grafičkim procesorima (engl. *Graphics Proccesing Unit*, GPU). Time je omogućeno učinkovitije rješavanje mnogih složenijih računalnih problema na grafičkim procesorima nego što to omogućavaju procesori opće namjene (engl. *Central Processing Unit*, CPU). Danas CUDU koriste znanstvenici i istraživači na raznim područjima, kao što je: obrada slike i videa, istraživanja u biologiji i kemiji, dinamičkim simulacijama fluida, CT rekonstrukcije slike, seizmičke analize i mnoge druge [25].

Organizacija sklopovlja

Budući da su grafički procesori dizajnirani za drugačije tipove aplikacija u odnosu na procesore opće namjene, njihova se arhitektura razvijala u drugačijem smjeru. Grafički procesori sastoje se od nekoliko razina koje su organizirane hijerarhijski. Na prvoj razini nalaze se protočni multiprocesori (engl. streaming multiprocessors), odnosno računske jedinice. Svaki multiprocesor sastoji se od nekoliko protočnih procesora (engl. stream processors), koristi se još i naziv jezgra. Multiprocesori su organizirani prema arhitekturi SIMD (engl. Single Instruction Multiple Data). Temeljem toga slijedi bitna razlika između procesora opće namjene i grafičkih procesora. Jezgre procesora opće namjene od samog početka dizajnirane su da slijedno izvode niz instrukcija maksimalnom brzinom. Za razliku od toga, grafički procesor omogućava paralelnu obradu velike količine podataka. Jedna instrukcija obavlja se nad više podatkovnih elemenata, tj. nakon svakog procesorskog takta, svaki procesor multiprocesora obavlja istu instrukciju, ali nad različitim podacima. Valja primijetiti kako različiti multiprocesori mogu paralelno izvoditi različite naredbe u istom periodu glavnog takta grafičkog procesora, dakle oni su međusobno nezavisni. Procesori unutar određenog multiprocesora paralelno izvode istu naredbu.
Potrebno je napomenuti da arhitektura samih grafičkih procesora ovisi o njihovoj seriji. Na Slika 36 prikazana je arhitektura grafičkog procesora novije generacije, GT200 [26].



Slika 36 - Organizacija sklopovlja GPU procesora

Ovaj grafički procesor sastoji se od trideset protočnih multiprocesora, a unutar svakog multiprocesora nalazi se osam protočnih procesora. Tri takva multiprocesora mogu se vidjeti na slici. Protočni procesori imaju mogućnost izvođenja operacija nad cjelobrojnim podacima jednostruke (32 bita) preciznosti (engl. *single-precision integer*) i operacija nad podacima s pomičnim zarezom jednostruke (32 bita) preciznosti (engl. *single-precision foating point*). Pored osam protočnih procesora, unutar svakog multiprocesora, nalaze se: zajednička pričuvna memorija za podatke i instrukcije, logička kontrolna jedinica, 16kB zajedničke memorije, dvije jedinice za specijalne funkcije (engl. *special function units,* SFU) i instrukcijska jedinica (engl. *multithreaded instruction unit*). Specijalne jedinice, SFU koriste se za obavljanje posebnih funkcija, kao što su operacije dvostruke preciznosti (64 bita) pomičnog zareza (engl. *double-precision foating-point*) i transcendentalne operacije, tj. računanje sinusa, kosinusa i slično. Instrukcijska jedinica ima posebnu namjenu, njezin zadatak je raspoređivanje instrukcija svim protočnim procesorima i specijalnim jedinicama za složene funkcije [27].

Programski model

Program koji se izvodi u okruženju CUDA sastoji se od dijelova koji se odvojeno izvode na procesoru opće namjene i grafičkom procesoru. Dio programa s vrlo malo podatkovnog paralelizma ili onaj dio gdje ga uopće nema, implementira se standardnim jezikom C. Prevodi se i izvodi se na procesoru opće namjene kao i svaki drugi običan program. S druge strane, dio koda koji posjeduje veliku količinu podatkovnog paralelizma implementira se jezikom C i njegovim CUDA proširenjima za označavanje paralelnih funkcija i pridružene podatkovne strukture. Funkcija za takvu paralelnu obradu podataka na grafičkom procesoru u daljem tekstu naziva se

funkcija jezgre [26]. Izvođenje programa započinje na procesoru opće namjene, a pozivom funkcija jezgre izvođenje se nastavlja na grafičkom procesoru (Slika 37).

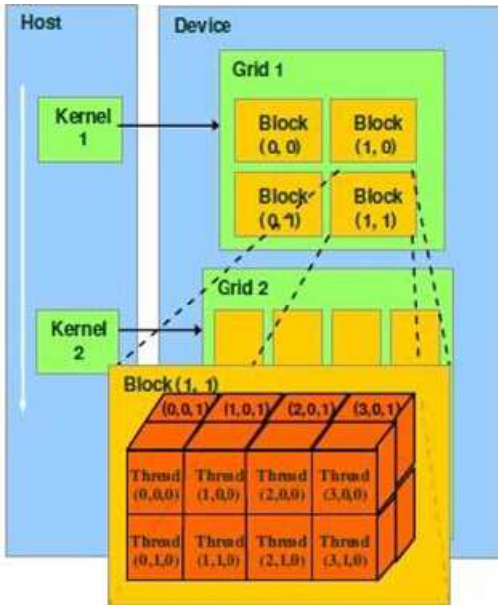
Slika 37 - Prinicip izvođenja koda (GPU/CPU)

Kako bi se iskoristila velika količina podatkovnog paralelizma, prilikom poziva funkcije jezgre generira se ogroman broj niti. Funkcija jezgre predstavlja dio koda koji će paralelno izvoditi sve niti. Generirane niti sačinjavaju mrežu niti (engl. *grid*). Kad sve niti generirane prilikom poziva funkcije jezgre izvrše, izvođenje se nastavlja na procesoru opće namjene. Programer je dužan prethodno zauzeti potrebnu količinu memorije GPU da bi se pozivom funkcije jezgre omogućilo izvođenja na grafičkom procesoru. Nakon toga, u zauzetu memoriju GPU prebacuju se svi podaci iz memorije *hosta* nužni za uspješno izvođenje funkcija jezgre. Ovdje se *hostom* smatra cjelina unutar koje se nalazi CPU. Nakon završetka izvođenja, potrebno je rezultat prebaciti natrag u memoriju *hosta*, te osloboditi prethodno zauzetu memoriju GPU.

Organizacija niti

Kao što je već ranije spomenuto, pozivom funkcije jezgre generira se mreža niti. Sve niti unutar mreže izvoditi će isti odsječak koda. Niti dohvaćaju različite podatke iz globalne memorije, nad kojima paralelno izvode istu instrukciju. Niti unutar mreže podijeljene su u blokove. Blokovi mogu biti jednodimenzionalni, dvodimenzionalni ili trodimenzionalni. Broj blokova u svakoj dimenziji i broj niti unutar jednog bloka određeni su ugrađenim varijablama koje je programer dužan navesti kod poziva funkcija jezgre. Njihova glavna namjena je omogućiti da svaki blok i svaka nit unutar tog bloka posjeduje jedinstvene koordinate pomoću kojih se pojedina nit razlikuje od svih ostalih. Osim njihovog međusobnog raspoznavanja, varijable se koriste i za određivanje pozicije podatka koji de pojedina nit obrađivati. Na Slika 38 prikazana je organizacija niti unutar četiri bloka. Kako svi blokovi sadrže jednak broj niti organiziranih na isti način, dovoljno je prikazati niti jednog bloka. Prikazani blok organiziran je kao trodimenzionalno 4×2×2 polje niti. U jednom bloku

nalazi se 16 niti, što daje ukupno 64 niti unutar mreže. U stvarnosti je taj ukupan broj niti daleko veći.



Slika 38 - Organizacija niti unutar četiri bloka

Potrebno je osigurati da sve niti unutar bloka završe s izvođenjem jedne faze prije nego što nastave s izvođenjem iduće faze. To se osigurava posebnom funkcijom za sinkronizaciju. Po pozivu te funkcije nit zaustavlja svoje izvođenje i čeka da preostale niti unutar bloka dođu do tog mjesta. Sinkronizacija niti između različitih blokova unutar mreže nije moguća. Mehanizam izvođenja instrukcija grupira niti jednog bloka u male skupine, tzv. osnove (engl. *wraps*). Broj niti unutar jedne osnove kod današnjih grafičkih procesora jednak je 32. Osnove se slijedno dodjeljuju multiprocesorima, gdje se sve niti te osnove istovremeno izvode na protočnim procesorima dodijeljenog multiprocesora. Prilikom raspodjele niti protočnim procesorima, protočni procesori će izvesti najmanje jednu instrukciju, nakon čega se mogu dodijeliti drugoj osnovi. Sve niti iste osnove istovremeno izvode istu operaciju. Ukoliko niti jedne osnove pristupaju podacima globalne memorije, za što je potrebno 400-600 ciklusa da bi adresirani podatak bio raspoloživ, za to se vrijeme izvodi se druga osnova. Nakon što resursi postanu raspoloživi, nastavit će se sa izvođenjem prve osnove. Time će se izbjeći memorijska latencija, a za to se brine mehanizam zasnovan na prioritetima. Maksimalna učinkovitost postiže se i paralelnim izvođenjem blokova. Ako grafički procesor ne posjeduje dovoljno resursa (multiprocesora), blokovi se mogu izvoditi jedan po jedan, izvođenje se ne mora odvijati istom brzinom. Svojstvo izvođenja iste aplikacije različitim brzinama naziva se transparentna skalabilnost i omogućava programerima lakše programiranje, te povećava iskoristivost aplikacija [27].

Organizacija memorije

Grafički procesori sadrže nekoliko tipova memorije koje su hijerarhijski organizirane. Neke od njih su već ranije spomenute, a ovdje je dan njihov detaljniji opis. Globalna memorija naziva se još i VRAM (engl. Video Random Access Memory), te predstavlja najveću količinu memorije koja je dostupna svim protočnim multiprocesorima. Količina globalne memorije varira od nekoliko stotina megabajta do reda veličine gigabajta, koliko nalazimo na najjačim grafičkim karticama (GTX295). Iznimka su integrirane grafičke kartice koje dijele memoriju sa središnjim procesorom. Pruža veliku propusnost (do 100GB/s), ali operacije pristupanja toj memoriji posjeduju veliku latenciju (nekoliko stotina ciklusa). Kako je vrijeme pristupanja ovoj memoriji vrlo sporo, protočni procesori dohvaćaju podatke u vlastite registre (privatna memorija) ili u dijeljenu memoriju i izvršavaju zadane operacije nad podacima. Po završetku obrade podataka, rezultati se ponovo upisuju u globalnu memoriju. Potrebno je spomenuti da je kod grafičkih kartica s podrškom većom ili jednakom Compute Capability 1.2 globalna memorija podijeljena u segmente od 32, 64 i 128 bajta koji su poravnati na adresama istih višekratnika. Prilikom dohvaćanja podataka u globalnu memoriju prvi podatak se sprema na adresu koja je višekratnik željenog segmenta. Programer treba voditi računa o tome jer broj memorijskih transakcija ovisi o broju pristupanja niti iste osnove različitim segmentima. Točnije, koliko niti iste osnove pristupa različitim segmentima toliko će biti memorijskih transakcija, bez obzira što se radi o slijednim adresama [26], [27].
Dijeljena memorija, koristi se još i naziv zajednička memorija. Radi se o izuzetno brzoj memoriji malog kapaciteta. Kod većine današnjih grafičkih procesora ona iznosi oko 16kB, ali uglavnom to ovisi o seriji GPU. Nalazi se na razini multiprocesora, a svaki multiprocesor posjeduje svoju dijeljenu memoriju kojoj mogu pristupati svi procesori tog multiprocesora. Pristupanje dijeljenoj memoriji drugog multiprocesora nije moguće. Protočni procesori mogu čitati i pisati podatke kao i kod globalne memorije. Pristupanje
ovoj memoriji je oko 150 puta brže u odnosu na globalnu memoriju. Zbog toga ona ima ulogu priručne memorije (engl. cache) procesora opće namjene. U nju se dohvaćaju svi podaci koji se često koriste i tako se ubrzava izvođenje programa. Razlika u odnosu na procesore opće namjene je da se programer brine o dohvaćanju podataka u dijeljenu memoriju, dok je to kod procesora opće namjene riješeno na sklopovskoj razini. To zahtjeva od programera veliko iskustvo i dobro poznavanje organizacije memorije jer u suprotnom nede doći do iskorištavanja brzine koju pruža dijeljena memorija.
Lokalna memorija je mala količina memorije na razini protočnog procesora. Svaki protočan procesor sadrži vlastitu lokalnu memoriju kojoj može pristupati. Pristupanje lokalnoj memoriji drugog protočnog procesora, bez obzira da li je izvan ili

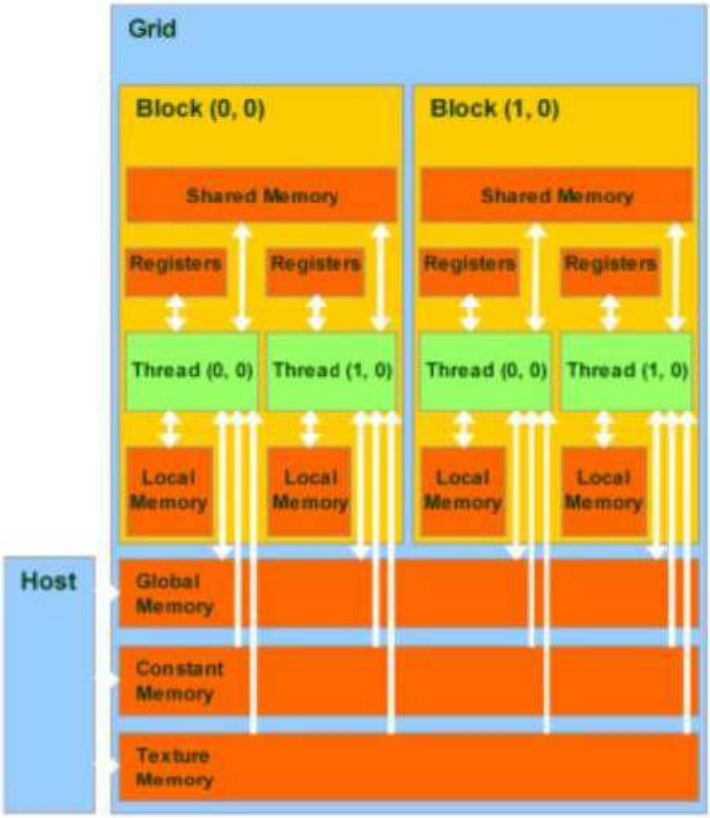
unutar multiprocesora, nije moguće. Kao i kod globalne memorije, pristupanje je ovoj memoriji izrazito sporo.

Privatna memorija naziv je za 32-bitne registre pridružene svakom protočnom procesoru. Jednako kao i kod lokalne memorije svaki protočan procesor može pristupati jedino vlastitom registru. Razlika je u brzini, pristupanje ovoj memoriji daleko je brže od pristupanja lokalnoj memoriji. Ovaj tip memorije najčešće se koristi za pohranu privatnih varijabli.

Memorija konstante je mala količina memorije veličine oko stotinu kilobajta. Podatke iz ove memorije mogu čitati svi multiprocesori, ali nemaju mogućnost ponovnog pisanja. Njezina uloga je slična priručnoj memoriji procesora opće namjene, a koristi se za pohranjivanje konstantnih vrijednosti. Pristupanje ovoj memoriji brže je od pristupanja globalnoj memoriji, no još je uvijek sporije od pristupanja dijeljenoj memoriji.

Memorija teksture definirana je na razini multiprocesora s ograničenim pristupom. To znači da svi multiprocesori mogu samo čitati podatke iz memorije teksture, dok njihovo mijenjanje ili upisivanje novih nije moguće. Memorija teksture se i dalje vrlo često koristi jer omogućava niz prednosti u odnosu na globalnu memoriju. Radi se o priručnoj memoriji iz koje se podaci daleko brže čitaju za razliku od globalne memorije. Nadalje, memorija teksture optimizirana je za 2D ili 3D prostorni lokalitet. Najbolje performanse postidi de niti iste osnove koje pristupaju adresama memorije teksture u neposrednoj blizini [26], [27]. Također, 8-bitni ili 16-bitni cjelobrojni podaci mogu se opcionalno konvertirati u 32-bitne podatke s pomičnim zarezom u rasponu *0.0 , 1.0+ ili [-1.0 , 1.0]. Razlog tome je što u mnogim slučajevima grafički procesor brže izvodi operacije u aritmetici sa pomičnim zarezom nego operacije s cijelim brojevima. Svi prethodno opisani tipovi memorije GPU prikazani su na Slika 39, kao i mogućnosti pristupanja od strane protočnog procesora i/ili multiprocesora.

Potrebno je napomenuti da lokalna, globalna, memorija za konstante i memorija teksture fizički čine jednu memoriju (VRAM), ali s različitim mogućnostima pristupa.



Slika 39 - Tipovi memorije GPU procesora

5.1 Obrada slike na GPU arhitekturi

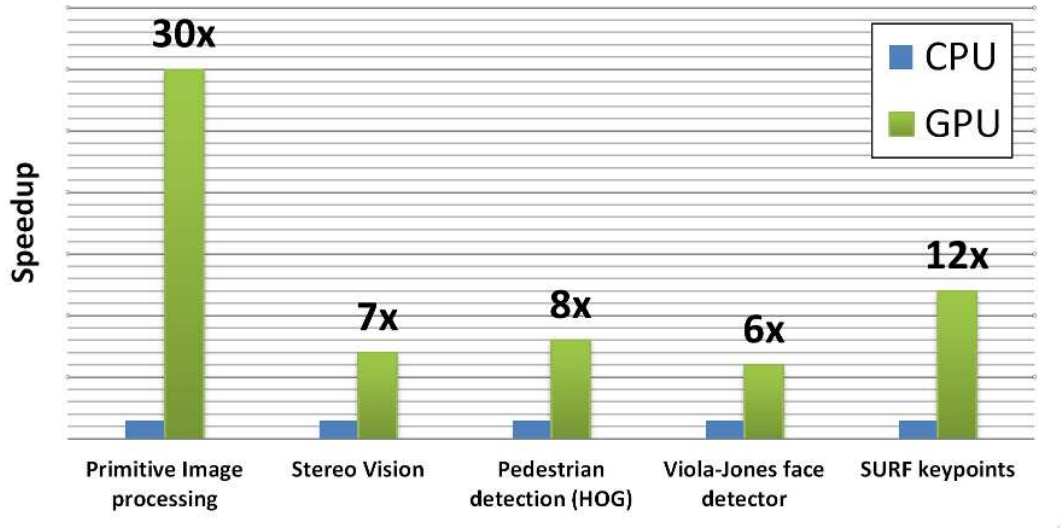
Kako je opisano u prethodnom poglavlju, GPU arhitektura je prikladna za rješavanje problema koji na standardnim CPU procesorima zahtijevaju više računalne snage. GPU arhitektura se sastoji od više manjih procesora koje rade paralelno te na taj način postižu bolje performanse od standardnih CPU-a koji rade sekvencionalno. Bitno je napomenuti da CPU i GPU arhitektura dolaze uvijek u kombinaciji, gdje CPU ima ulogu upravitelja, a GPU radnika.

Rješavanje problema i algoritama na GPU arhitekturi nameće podjelu istih na manje probleme te rješavanje tih manjih problema u paralelnom radu. Svaki algoritam je drugačiji i na korisniku je da ispravno razdijeli problem u manje probleme. Različita implementacija istog algoritma može imati različite performansne rezultate.

GPU arhitektura doživljava svoj napredak u trenutku kada CPU arhitektura dostiže svoja tehnološka ograničenja u smislu zagrijavanja i dimenzija.

Obrada slike i procesi sa slikama su dobar primjer problema koji se može usitniti. Od jednostavne primjene filtera nad slikom gdje se nad svakim pikselom vrši neka operacija pa do klasfikacijskih i detekcijskih algoritama koji se uzimaju dijelove slika i njene značajke.

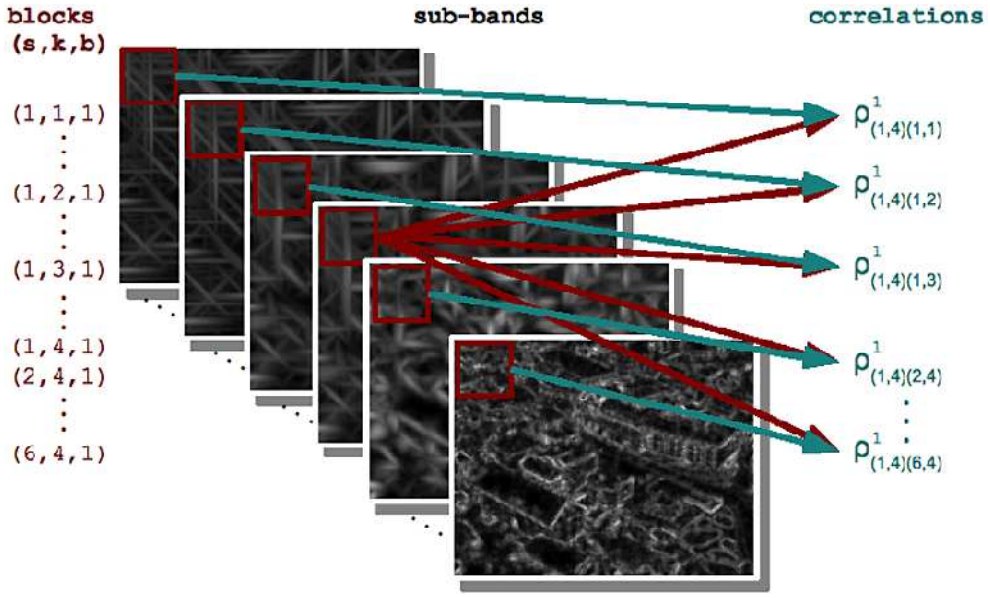
OpenCV biblioteka (biblioteka otvorenog koda) sadrži implementaciju osnovnih algoritama za obradu slike na GPU arhitekturi. Na Slika 40 je dana usporedba nekih algoritama za obradu slike.



Slika 40 - Usporedba nekih algoritama za obradu slike na CPU i GPU arhitekturi

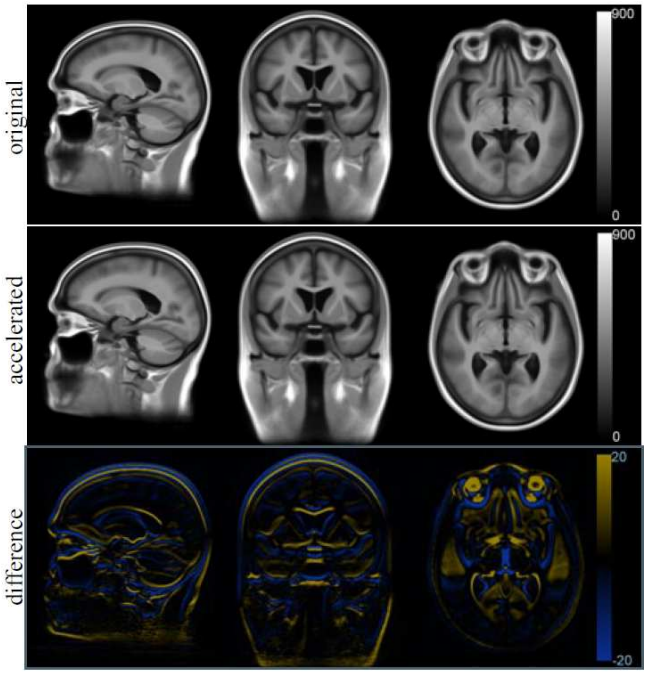
Klasifikacija slike se kod nadziranog učenja uvijek temelji na minimalno dvije faze: trening i test. U trening fazi algoritmu se predaju referentni primjeri klasa te algoritam "uči" za klasifikaciju slike u test fazi. Ako se test faza pokaže uspješna, algoritam se može primijeniti. Upravo je trening faza najčešće performansno zahtjevna. Ukoliko je klasifikacijski problem takav da su slike koje se koriste visokih rezolucija, tada je još opravdaniji zahtjev za implementacijom trening faze na GPU arhitekturi.

Autori [44] su primijenili klasifikacijski algoritam za slike dobivene iz zraka na GPU arhitekturi kako bi smanjili trajanje trening faze i na taj način dobili mogućnost lakše nadogradnje algoritma za učenje koji koriste. Navedeni algoritam dijeli sliku visoke rezolucije u manje slike te se svaka slika paralelno obrađuje na pojedinom bloku GPU arhitekture te joj se definira klasa. Prikaz je dan na Slika 41.



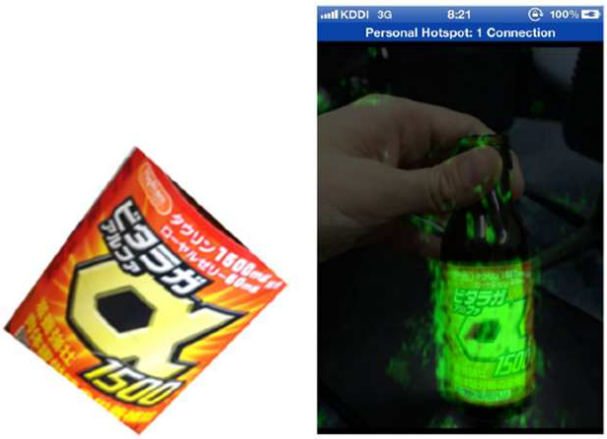
Slika 41 - Podjela slike visoke rezolucije u manje slike i distribucija na blokovima GPU arhitekture. Nad svakom manjom slikom vrši se izračun korelacije sa referentom slikom klase

Slike visoke rezolucije se često koriste u medicinskoj dijagnostici (na primjer slike magnetske rezonance). Obrada navedenih slika i područja slike može se izvršavati na GPU arhitekturi sa ciljem ubrzanja algoritma i dobivanja dijagnoze. Autori [45] u svom radu uvode optimizacijski algoritam koji registrira slike i 4 do 5 puta brže i kvalitetnije u odnosu na standardne CPU procese. Navedeni algoritam iskorištava činjenicu da GPU arhitektura može obraditi veću količinu informacija u manje vremena od uobičajene CPU arhitekture. Prikaz dobivenih razlika u dohvaćenim slikama magnetske rezonance dan je Slika 42.



Slika 42 - Prikaz razlika slika magnetske rezonance dobivenih CPU i GPU arhitekturom

Zahvaljujući razvoju mobilnih tehnologija, GPU procesori su danas dostupni i na mobilnim uređajima. Navedena činjenica otvara mogućnost primjene algoritama koji iskorištavaju GPU arhitekturu u realnom vremenu na mobilnim uređajima. Jedan takav primjer dali su autori [46] koji su primijenili algoritam za prepoznavanje uzoraka na mobilnom pametnom uređaju. Navedeni algoritam koristi kombinaciju modificiranog Random Fern algoritma (opisanog u poglavlju 3.1) i algoritma potpornih vektora (opisanih u poglavlju 3.2). Prikaz detekcije uzorka dan je na Slika 43.



Slika 43 - Detekcija uzoraka na pametnom telefonu koristeći GPU arhitekturu

6. Zaključak

Iako je područje obrade i klasifikacije dokumenta istraživano područje (Poglavlje 4.5), razvojem tehnologije i novim saznanjima u obradi slike, redovito ostaje prostora za nove metode klasifikacije.

Konstantan razvoj i napredak bilježi i polje računalnog vida koje se bavi detekcijom i opisom značajki slike. Od SIFT algoritma (Lowe - 1999) i HOG detektora (Dalal, Triggs et al. - 2005) koji su robusni i pouzdani u primjeni, do novijih detektora značajki FREAK (na primjer Fast Retina Keypoint - Alahi et al. - 2012) koji još nisu primijenjeni u realnim uvjetima, ali pokazuju bolje performanse izvođenja uz jednaku robusnost. Napredak u performansama izvođenja prilikom detekcije i opisa značajki dovodi do općeg poboljšanja performansi već postojećih algoritama za strojno učenje (na primjer stroj sa potpornim vektorima ili neuronske mreže) koji imaju dokazanu pouzdanost i primjenu.

U Poglavlju 6. navedeno je nekoliko rješenja koja koriste pečate kao značajke dokumenta prilikom klasifikacije kao i nekoliko rješenja koji koriste različite metode detektiranja predložaka pojedinog dokumenta sa ciljem klasifikacije ili barem kategorizacije dokumenta.

Standardni algoritmi za klasifikaciju dokumenta koriste već poznatu metodu očitavanja teksta u dokumentima (engl. Optical character recognition - OCR) te zatim koriste analizu teksta (dubinska analiza, mapiranje ili slično) za dobivanje podataka o tipu dokumenta. Ovakav pristup je opravdan za dokumente dobre kvalitete i "čitljivosti", ali na degradiranim dokumentima OCR zbog manjka informacija ima značajan postotak pogreške. Dodatnim poboljšanjem OCR metode nad takvim dokumentima povećava se kompleksnost sustava što u sustavima sa velikom količinom dokumenata može biti neprihvatljivo.

Izazovi postoje na području obrade dokumenata na paralelnim arhitekturama koje pružaju znatno bolje performanse izvođenja algoritama za detekciju i ekstrakciju značajki i do trenutka pisanja ovog rada, nije poznato niti jedno istraživanje koje na paralelnoj arhitekturi koristi neki od algoritama navedenih u Poglavlju 4 sa ciljem klasifikacije dokumenata.

7. Literatura

[1] "IBM What is big data? — Bringing big data to the enterprise". www.ibm.com. Retrieved 2013-08-26

[2] Priyadharshini N, Vijaya MS: Genetic Programming for Document Segmentation and Region Classification Using Discipulus, (IJARAI) International Journal of Advanced Research in Artificial Intelligence, Vol. 2, No. 2, 2013

[3] Priyadharshini N, Vijaya MS: Document Segmentation And Region Classification Using Multilayer Perceptron, IJCSI International Journal of Computer Science Issues, Vol. 10, Issue 2, No 1, 2013

[4] ShwetaMayor, Bhasker Pant Document Classification Using Support Vector Machine , International Journal of Engineering Science and Technology (IJEST), 4(4), 1741 - 1745., 2012

[5] Dick de Ridder, Robert P.W. Duin, Michael Egmont-Petersen, Lucas J. van Vliet, Piet W. Verbeek: Nonlinear image processing using artificial neural networks, 2003

[6] Hopfield, J. (1982). Neural networks and physical systems with emergent collective computational abilities. Proceedings of the National Academy of Sciences of the U.S.A. 81, 3088–3092.

[7] Hinton, G., Sejnowski, T., and Ackley, D. (1984). Boltzmann machines: constraint satisfaction networks that learn. Technical Report Report CMU-CS-84-119, Carnegie-Mellon University, Pittsburgh, PA.

[8] Wikipedia: http://hr.wikipedia.org/wiki/Umjetna_neuronska_mre%C5%BEa, 2014.

[9] Hertz, J., Krogh, A., and Palmer, R. G. (1991). Introduction to the theory of neural computation. Addison-Wesley, Reading, MA.

[10] Rumelhart, D. E., Hinton, G. E., andWilliams, R. J. (1986). Learning internal representations by error propagation. In Rumelhart, D.and McClelland, J.,editors, Parallel Distributed Processing: Explorations in the Microstructure of Cognition, volume I, pages 319–362. MIT Press, Cambridge, MA.

[11] Richard, M. D. and Lippmann, R. P. (1991). Neural network classifiers estimate Bayesian posterior probabilities.Neural Computation.3(4), 461–483.

[12] R. Maree et al.: A comparison of Generic Machine Learning Algorithms for Image Classification, 2013.

[13] David Capel: Random Forests and Ferns, Computer Vision Laboratory, PennState, Laboratory for Perception, Action and Cognition (LPAC), 2009 Seminars. (http://vision.cse.psu.edu/seminars/talks/2009/random_tff/ForestsAndFernsTalk.pdf)

[14] Alan Sambol, Prepoznavanje objekata klasifikacijom histograma orijentacije gradijenta strojem s potpornim vektorima, FER, 2010

[15] Fletcher, Support Vector Machines Explained, 2009. http://www.tristanfletcher.co.uk/SVM%20Explained.

[16] Greg Hamerly, Charles Elkan, Learning the k in k-means, 2003, http://books.nips.cc/papers/files/nips16/NIPS2003_AA36.pdf
[17] Fei-Fei Li., QuickReviews on ObjectRecognitionandBag-of-Words (BoW) Models, 2011., http://sensblogs.wordpress.com/2011/08/23/quick-reviews-on-object-recognition-and-bag-of-words-bow-models-by-fei-fei-li/
[18] Christopher Evans, Notes on theOpenSURFLibrary, Siječanj 2009. - http://opensurf1.googlecode.com/files/OpenSURF.pdf
[19] Badgerati, Computer Vision – The Integral Image, 2010, http://computersciencesource.wordpress.com/2010/09/03/computer-vision-the-integral-image
[20] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. European Conference on Computer Vision, 2006.
[21] Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, 2005.
[22] M. Özuysal, M. Calonder, V. Lepetit, P. Fua, Fast Keypoint Recognition using Random Ferns, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 32, Nr. 3, pp. 448 - 461, ožujak 2010.
[23] Lowe, D.G. Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision. 60 (2004), 91–110
[24] Vedaldi A. SIFT Tutorial, VLFeat: An open source computer vision in C, 2005, http://www.vlfeat.org/overview/sift.html, 1.03.2011.
[25] Živković O. Programska okruženja za programiranje na grafičkim procesorima, Seminar, Fakultet elektrotehnike i računarstva, 2010
[26] Kirk, D. i Hwu, W. CUDA Programming Model, 2006-2008., Chapter 2: CUDA Programming Model, http://sites.google.com/site/cudaiap2009/materials-1/cuda-textbook, 15.04.2011.
[27] Trbojević, A. Izvedba algoritama računalnog vida na grafičkim procesorima, Završni rad, Fakultet elektrotehnike i računarstva, 2010.
[28] A robust stamp detection framework on degraded documents -Guangyu Zhu, Stefan Jaeger, David Doermann SPIE Conference on Document Recognition and Retrieval, 2006
[29] P.V.C. Hough, Method and means for recognizing complex patterns, U.S. Patent 3069654, 1962.
[30] S. Tsuji, F. Matsumoto, “Detection of ellipses by a modified Hough transformation,” IEEE Trans. Comput., vol. 27, no. 8, pp.777–781, 1978.
[31] H. Li, M.A. Lavin, R. J. Le Master, “Fast Hough transform: a hierachical approach,” J. Comput. Vision Graphics Image Process., vol. 36, pp.139-161, 1986.
[32] R. A. McLaughlin, “Randomized Hough transform: improved ellipse detection with comparison,” Pattern Recognition Lett., vol. 19, no. 3-4, pp.299-305, 1998.
[33] Y. C. Cheng, S. Lee, “A new method for quadratic curve detection using K-RANSAC with acceleration techniques,” Pattern Recognition, vol. 28, no. 5, pp. 663-682, 1995.
[34] R. Krishnapuram, O. Nasraoui, H. Frigui, “The Fuzzy C spherical shells algorithms: a new approach,” IEEE Trans. Neural Networks, vol. 3, no. 5, pp. 663-671, 1992.

[35] R. N. Dave, “Generalized fuzzy C-shells clustering and detection of circular and elliptical boundaries,” Pattern Recognition, vol. 25, no.7, pp. 713-721, 1992.

[36] A. Fitzgibbon, M. Pilu, R. B. Fisher, “Direct least square 2tting of ellipses,” IEEE Trans. Pattern Anal. Machine Intell., vol. 2l, no. 5, pp. 477-480, 1999.

[37] P. Yin, “A new circle/ellipse detector using genetic algorithms,” in Proc. IPPR on CVGIP Taipei, pp. 362-368, 1998.

[38] Paweł Forczmański, Dariusz Frejlichowski, Robust Stamps Detection and Classification by Means of General Shape Analysis, Computer Vision and Graphics, Lecture Notes in Computer Science Volume 6374, 2010, pp 360-367, 2010

[39] Dariusz Frejlichowski, Paweł Forczmański: General shape analysis applied to stamps retrieval from scanned documents, Proceedings of the 14th international conference on Artificial intelligence: methodology, systems, and applications, Pages 251-260, 2010

[40] Daniel Esser, Daniel Schuster, Klemens Muthmann, Michael Berger, and Alexander Schill:Automatic indexing of scanned documents: a layout-based approach. *DRR, volume 8297 of SPIE Proceedings, SPIE,* (*2012*)

[41] Jianying Hu, Ramanujan Kashi, Gordon Wilfong: Comparison and Classification of Documents Based on Layout Similarity, Information Retrieval, May 2000, Volume 2, Issue 2-3, pp 227-243

[42] Priyadharshini, Vijaya: Document Segmentation And Region Classification Using Multilayer Perceptron, IJCSI International Journal of Computer Science Issues, Vol. 10, Issue 2, No 1, March 2013

[43] E. Appiani, F. Cesarini, A.M. Colla, M. Diligenti, M. Gori, S. Marinai, G. Soda: Automatic document classification and indexing in high-volume applications, International Journal on Document Analysis and Recognition December 2001, Volume 4, Issue 2, pp 69-83

[44] Rok Češnovar, Vladimir Risojević, Zdenka Babić, Tomaž Dobravec , Patricio Bulić: A GPU implementation of a structural-similarity-based aerial-image classification, The Journal of Supercomputing, 2013, Volume 65, Issue 2, pp 978-996

[45] Denis P Shamonin, Esther E Bron, Boudewijn P.F. Lelieveldt, Marion Smits, Stefan Klein and Marius Staring: Fast Parallel Image Registration on CPU and GPU for Diagnostic Classification of Alzheimer's Disease, Frontiers in Neuroinformatics, ISSN: 1662-5196, 2013

[46] Vsevolod Yugov and Itsuo Kumazawa: Combination of SVM and FERN for GPU-assisted texture recognition on mobile devices, Advances in Computer Science: an International Journal, Vol. 2, Issue 3, 2013