SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 692

Izvedba biblioteke za strojno očitanje rukom popunjavanih sedamsegmentnih znamenki

Melita Kokot

Zagreb, lipanj 2014.

Zahvala mentoru prof. dr. sc. Siniši Šegviću i dr. sc. Marku Čupiću na stručnoj pomoći i prijedlozima pri izradi ovog diplomskog rada te pomoći kod prikupljanja testnog skupa slika.

SADRŽAJ

1. Uvod

Strojno očitavanje ispitnih obrazaca ubrzava i olakšava proces ispravljanja ispita s pitanjima s višestrukim ponuđenim odgovorima. Razmatra se rješavanje problema identifikacije pristupnika očitavanjem njegovog matičnog broja iz niza popunjenih sedamsegmentnih znamenaka. Razvijen je sustav raspoznavanja znamenaka iz skupa sedam praznih uokvirenih polja raspoređenih tako da formiraju broj osam. Pravilnim zacrnjivanjem polja na taj je način moguće oblikovati bilo koju arapsku znamenku. Pristupnik bi trebao zacrniti polja tako da ona predstavljaju niz brojeva koji ga identificiraju, npr. jedinstveni matični broj akademskog građanina (JMBAG), a računalo bi nakon obrade skenirane slike trebalo raspoznati broj koji je korisnik ispunio.

U sklopu predmeta Završni rad razvijena je osnovna inačica sustava za raspoznavanje matičnog broja na temelju skenirane slike zacrnjenih sedamsegmentnih znamenaka. Razvijeni sustav nije bio dovoljno robustan i precizan za stvarnu primjenu. U ovom radu opisana je poboljšana inačica tog sustava s drugačijim rješenjima za određene probleme.

Sustav u jednom trenutku obrađuje i analizira jednu sliku i na temelju izračunatih parametara daje rezultat za tu ulaznu sliku. Sustav može raspoznati bilo koji broj znamenaka, a taj broj može se podesiti preko argumenata naredbenog retka, ili se koristi pretpostavljena vrijednost 10.

U tekstu su najprije opisane metode i algoritmi predobrade slike, određivanja geometrijskih svojstava znamenki te način raspoznavanja pojedine znamenke. Nakon toga opisuje se skup izvornih slika na kojemu je obavljeno testiranje uz primjere dobrog i lošeg popunjavanja obrazaca. Zatim je predstavljena programska izvedba opisanog rješenja u programskom jeziku C++ te je na kraju dana analiza rezultata testiranja ulaznog skupa slika i usporeba uspješnosti stare i nove inačice postupka.

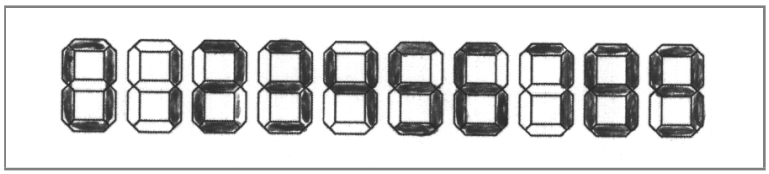
2. Korišteni algoritmi i metode

2.1. Predobrada slike

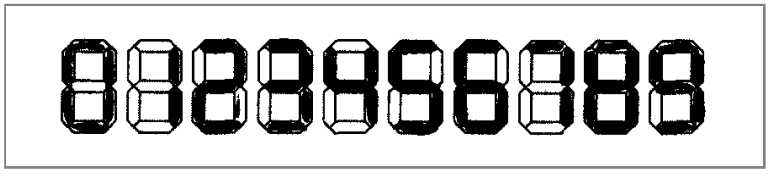
Prvi korak u gotovo svakom postupku analize slike je predobrada ulazne slike kako bi se olakšalo dobivanje željenih informacija iz slike. Predobrada se u ovom slučaju sastoji od binarizacije slike, odnosno razdvajanje objekta prvog plana od pozadine, traženja osi orijentacije objekta te, ukoliko je potrebno, rotacije slike. Ulazna slika je siva slika s vrijednostima slikovnih elemenata u rasponu od 0 do 255.

2.1.1. Binarizacija

Binarizacija slike je uobičajena i jednostavna metoda predobrade slike. Svaki slikovni element ulazne slike uspoređuje se sa zadanim pragom te se na temelju toga određuje vrijednost tog slikovnog elementa u izlaznoj slici. Za slikovne elemente ulazne slike čija je vrijednost veća ili jednaka od praga, odgovarajućim slikovnim elementima izlazne slike pridružuje se 0 (crni slikovni elementi, pozadina), dok se onima s vrijednošću manjom od praga pridružuje 1 (bijeli slikovni elementi, prednji plan). Sve ilustracije u ovom radu prikazane su invertirano (prednji plan je obojan crno) radi uštede kod tiska samog rada.



Slika 2.1: Izvorna slika



Slika 2.2: Binarna slika

Algoritam binarizacije primijenjen u ovom slučaju je globalan, tj. za cijelu sliku koristi se isti prag. Za određivanje praga binarizacije isprobane su dvije metode:

- algoritam uravnotežavanja histograma (engl. *Balanced Histogram Thresholding, BHT*),
- vršna metoda (engl. *Mode method*).

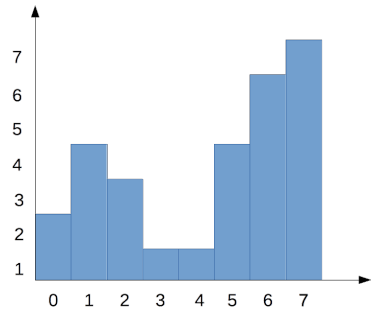
Za obje metode potrebno je najprije izračunati histogram sive slike, odnosno slike s 256 razina sive boje. Histogram je grafički prikaz koji predstavlja raspodjelu slikovnih elemenata po svih 256 sivih razina. Na horizontalnoj osi nalaze se pretinci (engl. *bins*) s vrijednostima intenziteta 0-255, a na vertikalnoj osi broj slikovnih elemenata koji pripadaju odgovarajućem pretincu.

BHT

Algoritam uravnotežavanja histograma (engl. *Balanced Histogram Thresholding, BHT*) pokušava pronaći granicu koja dijeli histogram na dva dijela približno jednakih težina. Metoda *važe* histogram, provjerava koja strana je teža, oduzima pretince s teže strane tako dugo dok ne postane lakša. Postupak se ponavlja sve dok ne ostane samo jedan pretinac koji predstavlja pronađeni prag.

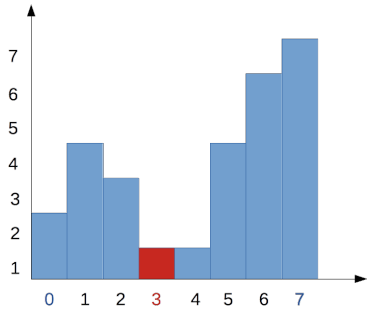
Kao početna granica kod BHT-a uzima se sredina horizontalne osi, odnosno pretinac 127. Zatim se računaju težine histograma lijevo i desno od granice kao zbroj svih pretinaca histograma na toj polovici. Od težine teže strane oduzima se vrijednost krajnjeg pretinca (najdesniji na desnoj polovici histograma, najljeviji na lijevoj), a granica se po potrebi pomiče za jedan pretinac prema suprotnoj strani, tako da uvijek bude na sredini preostalih pretinaca.

Slijedi primjer rada algoritma na jednostavnom histogramu:



Slika 2.3: Jednostavan histogram

Početno stanje:



$pocetak = 0$

$kraj = 7$

$prag = 3$

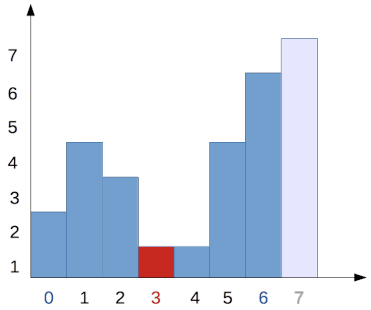
$tezina\_lijevo = 2 + 4 + 3 + 1 = 10$

$tezina\_desno = 1 + 4 + 6 + 7 = 18$

Slika 2.4: BHT - početak

Početak je prvi pretinac s lijeve strane čija se vrijednost ubraja u težinu lijeve polovice histograma, a kraj zadnji pretinac s desne strane. Vrijednost pretinca praga ubrojena je u težinu lijeve polovice histograma.

1.korak:



$tezina\_lijevo < tezina\_desno$:

$tezina\_desno = tezina\_desno - pretinac[7] = 18 - 7 = 11$

$pocetak = 0$

$kraj = 6$

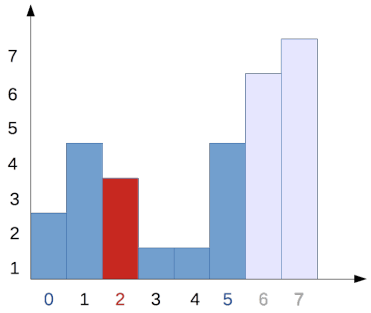
$prag = 3$

$tezina\_lijevo = 10$

Slika 2.5: BHT - korak 1

Pošto je desna polovica histograma teža, od nje se oduzima vrijednost zadnjeg pretinca s desne strane, te se taj pretinac odbacuje za sljedeću iteraciju algoritma. Prag je još uvijek na sredini preostalih pretinaca pa se ne treba pomaknuti za jedno mjesto ulijevo.

2. korak:



Slika 2.6: BHT - korak 2

$tezina\_lijevo < tezina\_desno$:

$tezina\_desno = tezina\_desno - pretinac[6] = 11 - 6 = 5$

$pocetak = 0$

$kraj = 5$

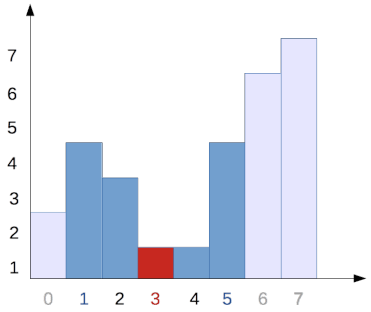
$prag = 2$

$tezina\_lijevo = tezina\_lijevo - pretinac[3] = 10 - 1 = 9$

$tezina\_desno = tezina\_desno + pretinac[3] = 5 + 1 = 6$

Nakon odbacivanja 6. pretinca, prag više nije na sredini preostalih pretinaca pa se pomiče za jedno mjesto ulijevo. Potrebno je i osvježiti obje težine zbog pomicanja praga, prethodna vrijednost pretinca se pribraja desnoj težini, a oduzima se lijevoj.

3. korak:



Slika 2.7: BHT - korak 3

$tezina\_lijevo > tezina\_desno$:

$tezina\_lijevo = tezina\_lijevo - pretinac[0] = 9 - 2 = 7$

$pocetak = 1$

$kraj = 5$

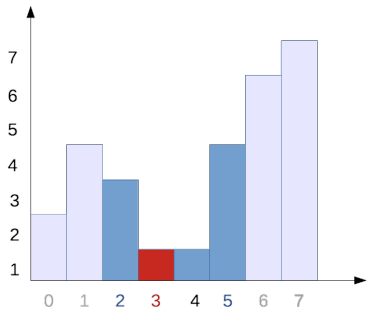
$prag = 3$

$tezina\_lijevo = tezina\_lijevo + pretinac[2] = 7 + 1 = 8$

$tezina\_desno = tezina\_desno - pretinac[2] = 6 - 1 = 5$

Sada je lijeva strana teža pa se odbacuje prvi pretinac s lijeve strane.

4. korak:



Slika 2.8: BHT - korak 4

$tezina\_lijevo > tezina\_desno$:

$tezina\_lijevo = tezina\_lijevo - pretinac[1] = 8 - 4 = 4$

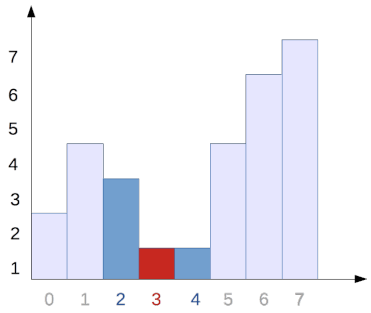
$pocetak = 2$

$kraj = 5$

$prag = 3$

$tezina\_desno = 5$

5. korak:



Slika 2.9: BHT - korak 5

$tezina\_lijevo < tezina\_desno$:

$tezina\_desno = tezina\_desno - pretinac[5] = 5 - 4 = 1$

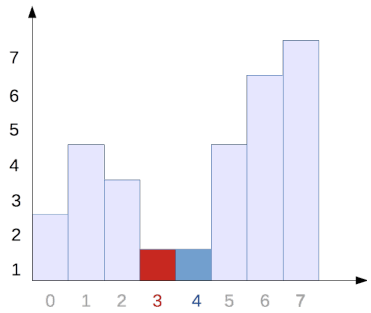
$pocetak = 2$

$kraj = 4$

$prag = 3$

$tezina\_lijevo = 4$

6. korak:



Slika 2.10: BHT - korak 6

$tezina\_lijevo > tezina\_desno$:

$tezina\_lijevo = tezina\_lijevo - pretinac[2] = 4 - 3 = 1$

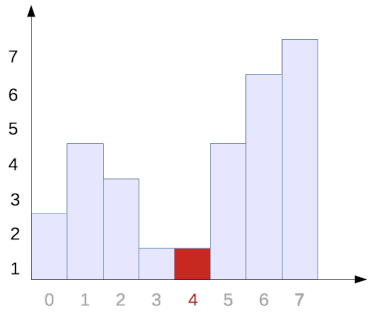
$pocetak = 3$

$kraj = 4$

$prag = 3$

$tezina\_desno = 1$

7. korak:



Slika 2.11: BHT - korak 7

$tezina\_lijevo == tezina\_desno$:

$tezina\_lijevo = tezina\_lijevo - pretinac[3] = 1 - 1 = 0$

$pocetak = 4$

$kraj = 4$

$prag = 4$

$tezina\_desno = tezina\_desno - pretinac[3] = 1 - 1 = 0$

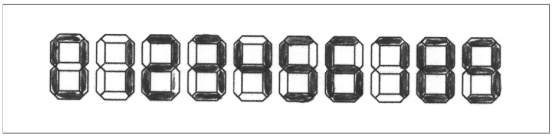
$tezina\_lijevo = tezina\_lijevo + pretinac[3] = 0 + 1 = 1$

Težine lijeve i desne polovice su jednake, odbacuje se preostali pretinac s lijeve strane (koji je ujedno i pretinac praga), a pošto je trenutna sredina 4 ($(pocetak+kraj)/2$), pomiče se i prag udesno te se osvježavaju težine. Kako su $pocetak$ i $kraj$ jednaki, tj. preostao je samo jedan pretinac, ovdje postupak završava.

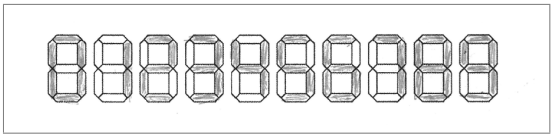
Kod slika znamenaka ispunjenih svjetlijim penkalama ili olovkama, histogram više nema jasno definiran brijeg na lijevoj strani koji bi trebao predstavljati objekte, dok je desni brijeg koji predstavlja bijelu pozadinu još uvijek velik. Kao posljedica toga, BHT metodom izračunava se preniski prag za binarizaciju jer je desna strana histograma kroz puno iteracija algoritma teža od lijeve strane.

Kao jednostavno rješenje za takve slike napravljena je mala modifikacija računanja praga BHT-om. Nakon što je izračunata granica za binarizaciju, provjerava se je li ona veća od minimalne dopuštene granice, koja je ovdje postavljena na 100. Ukoliko je izračunata granica manja od minimalne dopuštene, ona se ponovno računa, ali tako da se prvih 20 pretinaca s lijeve strane histograma odbaci. Postupak se ponavlja tako dugo dok granica nije veća od minimalne dopuštene, ali tako da se sada u svakoj iteraciji ignorira dodatnih 10 lijevih pretinaca. Ovaj postupak eliminira prvih nekoliko pretinaca koji su uobičajeno prazni ili imaju vrlo male vrijednosti, a početni prag je bliži desnom brijegu, pa su već kroz nekoliko iteracija algoritma pretinci desnog brijega odbačeni te se nakon toga prag pomiče lijevo - desno, a ne samo lijevo.

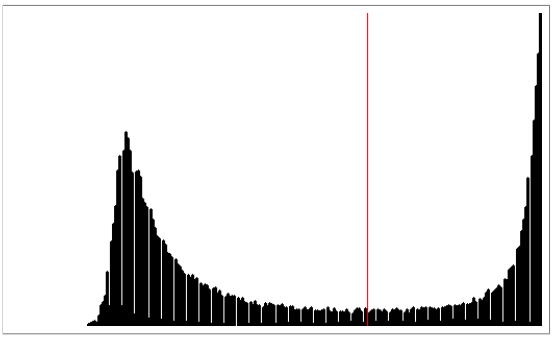
Međutim, kad su slike znamenaka ispunjene vrlo svijetlim olovkama, ova metoda ipak ne pronalazi optimalan prag zbog histograma s tri brijega, gdje prvi brijeg predstavlja crne rubove predloška znamenaka, drugi brijeg boju ispuna, a treći pozadinu. Drugi brijeg je uobičajeno dosta veći od prvog i vrlo blizu bijelog brijega što u ovom postupku znači putovanje praga cijelo vrijeme u lijevu stranu, a minimalni prag od 100 nije dovoljan za binarizaciju slike sa svijetlim objektom.



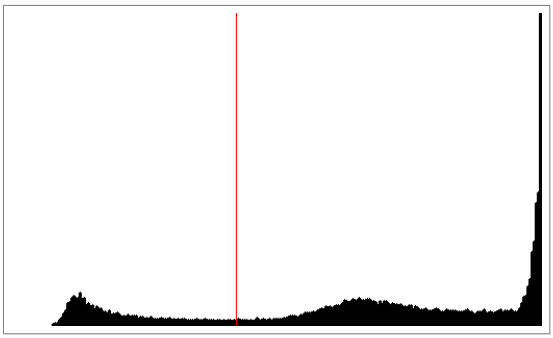
Slika 2.12: Originalna tamna slika



Slika 2.15: Originalna svijetla slika

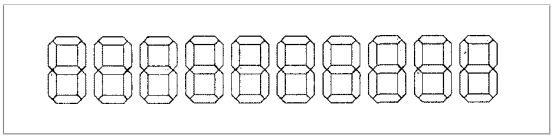


Slika 2.13: Histogram tamne s pragom



Slika 2.16: Histogram svijetle s pragom

Slika 2.14: Binarna slika tamne



Slika 2.17: Binarna slika svijetle

Lako se zaključuje da je metoda preosjetljiva na udio bijelih slikovnih elemenata te nije pogodna za ovu primjenu jer bi to zahtijevalo uvijek približno jednak omjer površine bijelog prostora oko znamenaka i površine znamenaka.

Vršna metoda

Vršna metoda pretpostavlja postojanje višemodalnog histograma i daje matematičku podlogu za izračun pragova za segmentaciju takve slike. Za izračun jednog praga k potrebno je pronaći par lokalnih maksimuma (i, j) u histogramu međusobno udaljenih za predefiniranu minimalnu udaljenost d, i najnižu točku k između njih. Ako su visine lokalnih maksimuma označene s $H(i)$ i $H(j)$, a visina lokalnog minimuma s $H(k)$, onda se šiljatost računa kao:

$$p = \frac{min(H(i), H(j))}{H(k)} \tag{2.1}$$

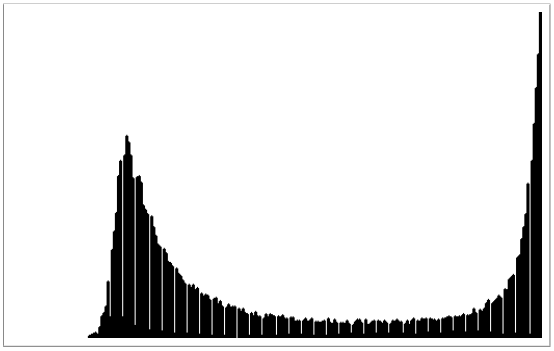
S obzirom na to da u histogramu ima više lokalnih maksimuma, algoritam je iterativan i zahtijeva variranje minimalne udaljenosti čime se iz histograma dobije niz trojki (i, j, k), a za prag se odabire k s najvećom šiljatošću. Ako je potrebno N pragova, uzima se N trojki (i, j, k) s najvećim šiljatostima. U ovom slučaju traže se 2

praga. Ako je histogram bimodalan, dovoljan je jedan prag (s najvećom šiljatosti) za uspješno odjeljivanje prednjeg plana od pozadine. Međutim, ako histogram ima 3 brijega (svijetlo ispunjene znamenke, tamni rubovi segmenata), prvi prag vrlo vjerojatno neće uspješno segmentirati sliku te je potrebno ponoviti proces sa sljedećim pragom (koji je između drugog i trećeg brijega). Uspješnost odjeljivanja objekta od pozadine se procjenjuje tijekom rada algoritma i dva su slučaja koja definiraju neuspješnost prvog praga:

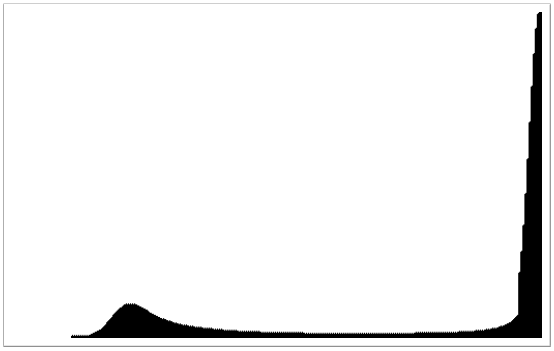
- ne uspijevaju se procijeniti visina i/ili širina znamenke zbog loših projekcija
- pouzdanost raspoznavanja niza znamenki je preniska (objašnjenje pouzdanosti u poglavlju 2.3.4).

Ako se dogodi bilo koji od dva navedena slučaja, cijeli postupak kreće ispočetka s drugim pragom.

Postupak pronalaženja pragova kreće sa zaglađivanjem histograma Gaussovim filtrom s parametrima $\mu = 0, \sigma = 5, P = 1$ (eksperimentalno odabrani) kako bi se smanjio broj lokalnih minimuma i maksimuma.



Slika 2.18: Histogram



Slika 2.19: Zaglađeni histogram

Histogram se zaglađuje konvolucijom s Gaussovom funkcijom:

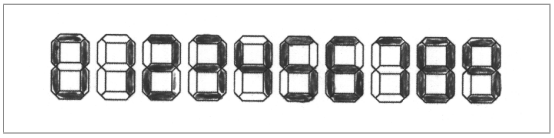
$$g(x) = \frac{1}{\sqrt{2\pi}\sigma} \cdot e^{\frac{-x^2}{2\sigma^2}} \tag{2.2}$$

Konvolucija se provodi u jednoj dimenziji, jer je i sam histogram predstavljen jednodimenzionalnim poljem, prema formuli:

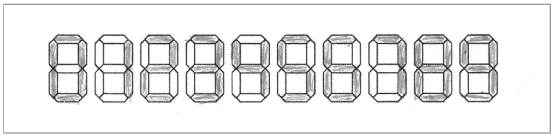
$$H[i] = \sum_{j=0}^{N-1} H[i + j - \frac{N-1}{2}] \cdot K[j] \tag{2.3}$$

Na zaglađenom histogramu traže se svi lokalni maksimumi i minimumi - ekspe-

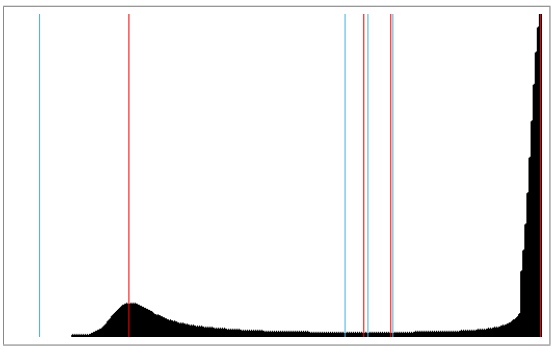
rimentalno je utvrđeno da nije potrebno ograničiti minimalnu udaljenost za traženje lokalnih maksimuma jer ih nema puno. Za svaki par lokalnih maksimuma traži se minimum između njih te se računa šiljatost prema 2.1. Dvije trojke (i, j, k) s najvećom šiljatošću daju dva tražena praga s vrijednostima k.



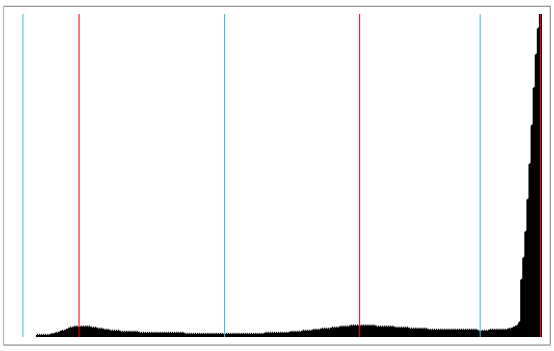
Slika 2.20: Originalna tamna slika



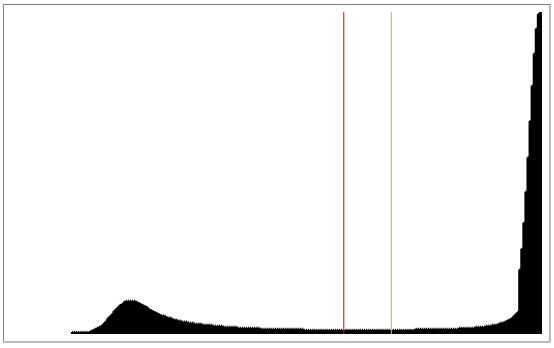
Slika 2.24: Originalna svijetla slika



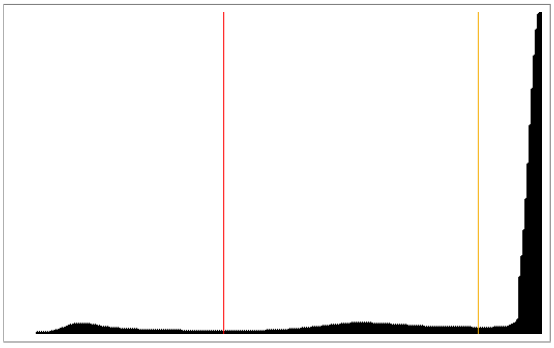
Slika 2.21: Histogram tamne s lokalnim ekstremima



Slika 2.25: Histogram svijetle s lokalnim ekstremima

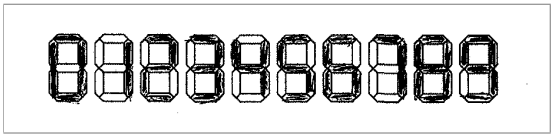


Slika 2.22: Histogram tamne s 2 praga



Slika 2.26: Histogram svijetle s 2 praga

Slika 2.23: Binarna slika tamne (prvi prag)



Slika 2.27: Binarna slika svijetle(drugi prag)

Na slici 2.21 prikazan je histogram niza znamenki sa slike 2.20 na kojemu se jasno vidi brijeg slikovnih elemenata objekta (lijevo) i brijeg slikovnih elemenata pozadine(desno) sa označenim lokalnim minimumima (plave linije) i lokalnim maksi-

mumima (crvene linije). Slika 2.22 prikazuje dva odabrana praga dobivena vršnom metodom, prvi prag označen je crvenom bojom, a drugi narančastom. U ovom slučaju za binarizaciju se uzima prvi prag, iako bi i binarizacija drugim pragom dala dobar rezultat jer je histogram bimodalan, a pragovi su slični po iznosu.

Predložak na slici 2.24 ispunjen je svijetlom olovkom što rezultira histogramom s tri brijega (slika 2.25). Pragovi vršne metode prikazani su na slici 2.26. Za ovakvu svijetlu sliku prvi prag bio bi loš odabir koji bi rezultirao samo s rubovima segmenata na binarnoj slici, kao što se može vidjeti na slici 2.17, a primjer binarizacije s drugim pragom vidljiv je na slici 2.27.

2.1.2. Orijentacija niza znamenki

Za analizu i dobivanje informacija iz slike znamenki, kao što su na primjer širina znamenke, visina znamenke ili širina razmaka između znamenki, promatrana nakupina slikovnih elemenata (u daljnjem tekstu objekt) mora biti uspravna. To znači da glavna os pružanja objekta mora biti paralelna s horizontalom. Potrebno je izračunati kut koji glavni pravac pružanja objekta zatvara s horizontalnim pravcem. Ako je taj kut zanemariv, slika se može smatrati uspravnom, dok je u suprotnom potrebno rotirati sliku.

Pravci pružanja objekta računaju se pomoću analize glavnih komponenata (engl. *Principal Component Analysis, PCA*), kako je opisano u nastavku.

Najprije je potrebno izračunati središnju točku objekta tako da se izračunaju aritmetičke sredine koordinata x i y za svaki slikovni element objekta:

$$\underline{\mathbf{m}} = \frac{1}{N} \sum_{i}^{N} \underline{\mathbf{g}}_i \tag{2.4}$$

gdje je $\underline{\mathbf{g}}_i$ x, odnosno y koordinata trenutnog slikovnog elementa, a N broj slikovnih elemenata objekta. Rješenje je točka (m_x, m_y) koja predstavlja središte objekta.
Zatim se računa matrica dimenzija $(2, N)$ koja će sadržavati odstupanja koordinata svakog slikovnog elementa od središta.

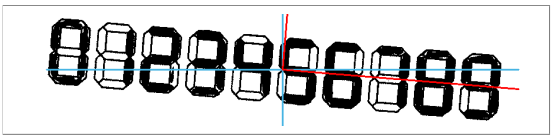
$$\mathbf{A} = \left[\underline{\mathbf{g}}'_i\right], \quad \underline{\mathbf{g}}'_i = \underline{\mathbf{g}}_i - \underline{\mathbf{m}} \tag{2.5}$$

U matrici $\mathbf{A}$ prvi redak odgovara odstupanju apscise, a drugi odstupanju ordinate. Sljedeći korak je izračunati matricu kovarijance koja će dati podatke o raspršenju slikovnih

elemenata objekta:

$$\mathbf{B} = \frac{1}{N}\mathbf{A} \cdot \mathbf{A}^T = \frac{1}{N}\sum_{i}^{N} \underline{\mathbf{g}}_i' \underline{\mathbf{g}}_i'^T \tag{2.6}$$

Nakon toga računaju se svojstveni vektori i svojstvene vrijednosti iz matrice kovarijance $\mathbf{B}$. Iz svojstvenih vektora lako se izračuna orijentacija objekta kao svojstveni vektor koji je bliži horizontalnoj osi.



Slika 2.28: Binarna s orijentacijom objekta

Slika 2.29: Rotirana binarna slika

2.1.3. Rotacija slike

Ukoliko je izračunati kut veći od maksimalne dopuštene vrijednosti kuta za uspravnu sliku, ulaznu sliku potrebno je rotirati. Korištena je interpolacijska metoda najbližeg susjeda (engl. *nearest neighbor interpolation*) i ekstrapolacija slikovnih elemenata kopiranjem postojećih rubnih točaka. Interpolacijska metoda najbližeg susjeda daje bolje rezultate od uobičajene bilinearne interpolacije (korištene u prvoj inačici postupka) i omogućava rotiranje direktno binarne slike jer ne računa nove vrijednosti slikovnih elemenata, već uzima vrijednosti susjednih slikovnih elemenata za trenutni slikovni element.

Za izračun rotacijske matrice potreban je kut θ dobiven na način opisan u prošlom poglavlju:

$$\mathbf{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{2.7}$$

Za svaki slikovni element izlazne slike $\underline{\mathrm{g}}'$ računaju se koordinate odgovarajućeg slikovnog elementa ulazne slike $\underline{\mathrm{g}}$ čija će vrijednost pripadati tom slikovnom elementu izlazne slike prema:

$$\underline{\mathbf{g}}' = \mathbf{R} \cdot \underline{\mathbf{g}} + T \tag{2.8}$$

gdje je T točka u koju se preslikava ishodište originalne slike. Odredišni slikovni elementi dobivaju se interpolacijskom metodom najbližeg susjeda.

2.2. Procjena geometrijskih parametara

Iz uspravne binarne slike niza sedamsegmentnih znamenki mogu se procijeniti sljedeći parametri predloška:

- širina znamenke
- visina znamenke
- razmak između znamenaka
- širina segmenta znamenke

Ti parametri se procjenjuju iz horizontalne i vertikalne projekcije binarne slike niza znamenki.

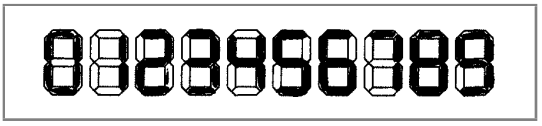
2.2.1. Projekcije

Horizontalna i vertikalna projekcija računaju se kao sume slikovnih elemenata objekta po retcima, odnosno stupcima.

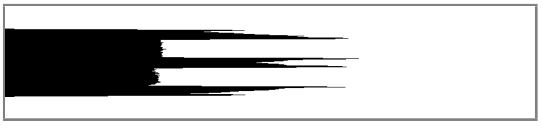
Horizontalna projekcija

Horizontalna projekcija je suma slikovnih elemenata po retcima:

$$H[i] = \sum_{i=0}^{m-1} X[i,j] \tag{2.9}$$



Slika 2.30: Binarna slika

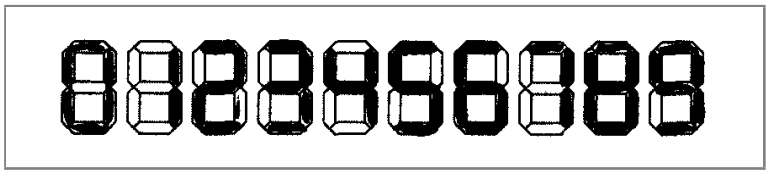


Slika 2.31: Horizontalna projekcija

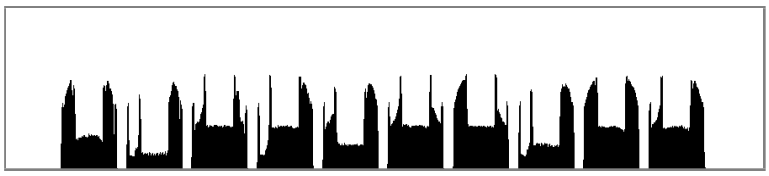
Vertikalna projekcija

Vertikalna projekcija je suma slikovnih elemenata po stupcima:

$$V[j] = \sum_{j=0}^{n-1} X[i,j] \tag{2.10}$$



Slika 2.32: Binarna slika



Slika 2.33: Vertikalna projekcija

2.2.2. Rubne točke znamenki

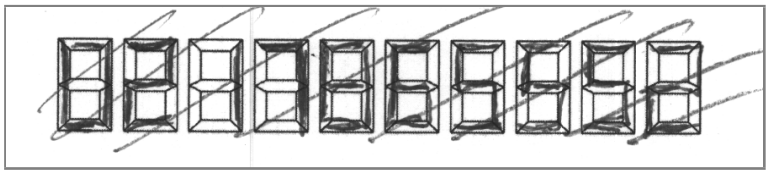
Za lociranje područja znamenki potrebno je pronaći rubne točke svake znamenke. Kada su poznate rubne točke, lako se izračunaju širina i visina te razmak između znamenki. Pritom se pretpostavlja da su sve znamenke početnog obrasca jednakih dimenzija s jednakim razmacima između njih te da su znamenke uspravne. Rubne točke računaju se iz projekcija.

y koordinate

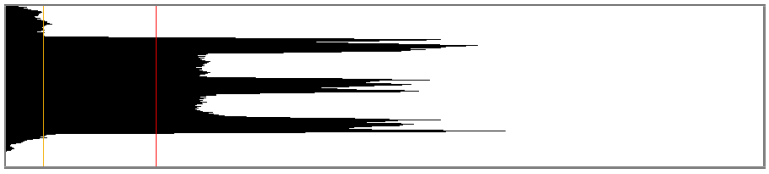
y koordinate niza znamenki računaju se iz horizontalne projekcije. Traži se samo jedna gornja i jedna donja y koordinata zbog pretpostavke o uspravnosti niza znamenki i jednakih dimenzija znamenki.

Najprije je potrebno odrediti dva praga $p1$ i $p2$ kako bi se što preciznije i robusnije našli indeksi horizontalne projekcije koji određuju tražene y koordinate. Prag $p1$ definiran je kao 30% iznosa najveće vrijednosti horizontalne projekcije, a prag $p2$ kao 25% od praga $p1$. Ovako odabrani pragovi otporni su kod određivanja koordinata na slikama sa slikovnim elementima objekta koji su malo izvan okvira predloška kao i na slučajne mrlje ili išarane linije izvan okvira predloška.

Traži se prvi element horizontalne projekcije veći od praga $p1$, a takav da je sljedećih barem 20 elemenata horizontalne projekcije veće od praga $p2$. Indeks tog elementa je gornja y koordinata niza znamenki. Slično se pronalazi i donja y koordinata za koju se traži zadnji element horizontalne projekcije veći od praga $p1$ kojemu prethodi barem 20 elemenata horizontalne projekcije koji su veći od praga $p2$.



Slika 2.34: Prošarana slika niza znamenki



Slika 2.35: Horizontalna projekcija s pragovima $p1$ (crveni, desno) i $p2$ (narančasti, lijevo)

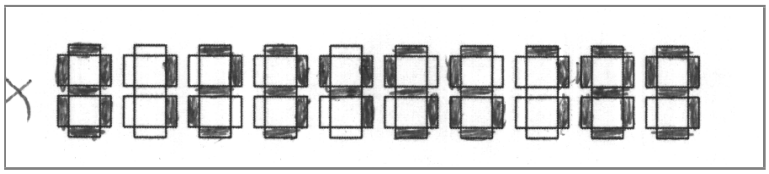
Slika 2.36: Horizontalna projekcija s y koordinatama (plavo)

Određivanje očekivane širine znamenke i razmaka

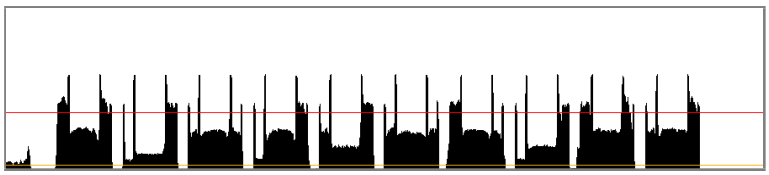
Pronalaženje točnih x koordinata svake znamenke nije jednostavno kao određivanje y koordinata. Razlog je to što su na horizontalnoj projekciji zbrojeni svi slikovni elementi objekta niza znamenki zajedno u jedan skup, dok su na vertikalnoj projekciji slikovni elementi svake znamenke zbrojeni odvojeno i međusobno su različiti. Također se vertikalne projekcije između različitih ulaznih slika više razlikuju nego horizontalne projekcije. Dodatno, postupak mora biti otporan i na malo šaranja izvan okvira predloška ili u prostoru između dvije znamenke. Stoga se prije traženja x koordinata znamenki određuju x koordinate početka i kraja niza znamenki, očekivana širina znamenke i razmak, a visina znamenke se može odrediti iz y koordinata, kako bi se tražene x koordinate što preciznije izračunale.

x koordinate početka i kraja niza znamenki računaju se na sličan način kao i y koordinate (2.2.2), uz prag $p1 = 0.6 * visina$ te $p2 = 0.05 * visina$, gdje je $visina$ jednaka razlici y koordinata niza znamenki. Kako bi se izbjegla pogrešna detekcija početka niza zbog, na primjer, slova *X* ispred niza znamenki, minimalan broj elemenata vertikalne projekcije koji trebaju biti veći od praga $p2$ je 10. Takvim elementima vertikalne projekcije može slijediti najviše 20 elemenata vertikalne projekcije s vrijednostima ispod praga $p2$. Ako takvih elemenata ima više, promatrani indeks vertikalne projekcije nije pravi početak niza znamenki. Analogno se određuje i x koordinata kraja

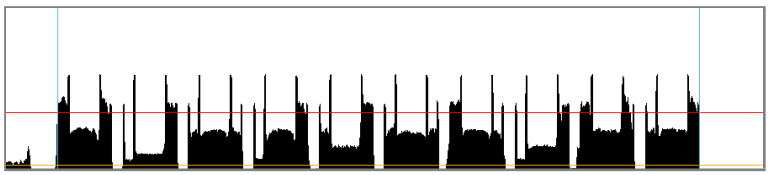
niza znamenki, samo s kraja vertikalne projekcije prema početku.



Slika 2.37: Slika niza znamenki s 'x'

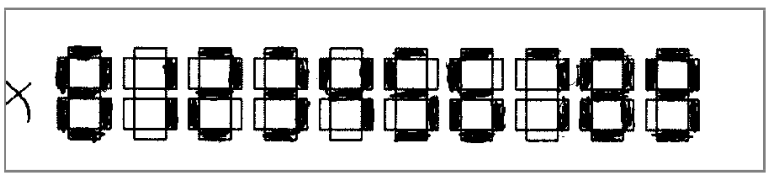


Slika 2.38: Vertikalna projekcija s pragovima $p1$ (crveni, gornji) i $p2$ (narančasti, donji)

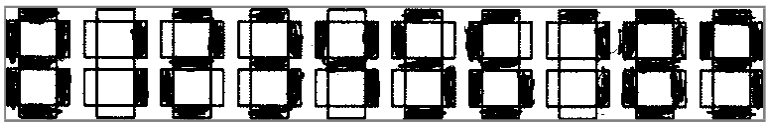


Slika 2.39: Horizontalna projekcija s x koordinatama početka i kraja niza (plavo)

Nakon što su poznate i krajnje x koordinate niza znamenki, radi lakšeg korištenja u daljnjim postupcima, izreže se dio slike koji sadrži samo niz znamenki, kao i pripadna vertikalna projekcija.



Slika 2.40: Binarna slika niza znamenki s 'x'



Slika 2.41: Izrezana binarna slika)

Širina niza znamenki računa se kao :

$$sirina\_niza = x_{kraj} - x_{pocetak} + 1, \tag{2.11}$$

očekivana širina znamenke računa se kao:

$$ocekivana\_sirina = \frac{sirina\_niza \cdot T}{N}, \tag{2.12}$$

očekivani razmak znamenke računa se kao:

$$ocekivani\_razmak = \frac{sirina\_niza - ocekivana\_sirina \cdot N}{N - 1}, \tag{2.13}$$

gdje je T postotak udjela širina svih znamenki između početka i kraja niza znamenki ($1 - T$ su razmaci), a N je broj znamenki u nizu. Oba parametra mogu se podesiti preko argumenata naredbenog retka, a pretpostavljene vrijednosti su $T = 0.85$ i $N = 10$.

x koordinate

Traženje x koordinata svih znamenaka odvija se na sličan način kao i traženje x koordinata početka i kraja niza. Prag $p1$ je 35% visine znamenke, a prag $p2$ četvrtina iznosa elementa vertikalne projekcije s indeksom na osmini duljine polja. Minimalna širina znamenke je 70% očekivane širine znamenke, a minimalni razmak je 3. Postupak se iterira kroz cijelu vertikalnu projekciju i u jednom prolazu se pronalaze samo x koordinate početaka znamenki. Iteriranjem od kraja polja vertikalne projekcije prema početku dobivaju se i x koordinate krajeva znamenki.

Ako se ovim postupkom prepoznalo manje od $N/2$ koordinata početaka ili krajeva, pretpostavlja se da je prag $p2$ prenizak te se diže za vrijednost 5 i postupak traženja x koordinata početaka i krajeva se ponavlja.

U sljedećem koraku provjerava se je li broj pronađenih x koordinati početaka jednak broju x koordinati krajeva. Ako je broj različit, provodi se "prisilno određivanje rubova" s očekivanom širinom znamenke i očekivanim razmakom.

```
float pocetak = 0;
for (int i = 0; i < N; ++i) {
  pocetne_tocke.push_back(pocetak);
  krajnje_tocke.push_back(pocetak + ocekivana_sirina);
  pocetak += ocekivana_sirina + ocekivan_razmak;
}
```

Nakon toga provjerava se točnost pronađenih x koordinata uz pomoć prosječne širine znamenke i prosječnog razmaka, koji se mogu izračunati na temelju x koordinata početaka i kraja. S obzirom na udaljenost između x koordinata početka i kraja pojedine znamenke, moguća su 4 scenarija:

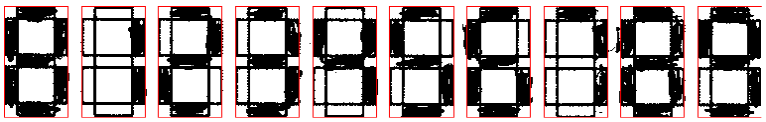
– udaljenost je manja od 30% širine znamenke - koordinate se brišu

– udaljenost trenutne i sljedeće znamenke je manja od 70% širine znamenke - ovo je znamenka razdvojena na dva dijela zbog previsokog praga $p2$; uzima se početna koordinata prve znamenke kao početak i krajnja koordinata druge kao kraj nove, spojene znamenke

– udaljenost je veća za 40% od širine znamenke - dvije ili više znamenaka spojenih u jednu zbog preniskog praga $p2$; na temelju omjera te udaljenosti i prosječne širine znamenki procjenjuje se koliko je spojenih znamenaka te se sukladno tome nalaze koordinate spojenih znamenaka

– udaljenost je približno jednaka prosječnoj širini znamenke - koordinate su u redu.

Ako i nakon ovog koraka broj x koordinata nije jednak broju znamenki N, cijeli postupak kreće od početka, ali s drugim pragom binarizacije koji se izračunat pomoću vršne metode. U suprotnom su pronađene sve koordinate rubnih točaka znamenaka i slika se može podijeliti na N podslika znamenaka.



Slika 2.42: Binarna slika s označenim regijama interesa

2.2.3. Razmak, širina, visina

Nakon što su poznate rubne točke svake znamenke, mogu se procijeniti razmak, širina i visina znamenke. Razmak između dvije znamenke određuje se prema:

$$r[i] = x\_pocetak[i+1] - x\_kraj[i] \tag{2.14}$$

gdje su $x\_pocetak$ i x_kraj x koordinate početka i kraja dobivene iz vertikalne projekcije. Kao konačna vrijednost prosječnog razmaka uzima se medijan od r.

Na sličan način računa se i širina znamenke:

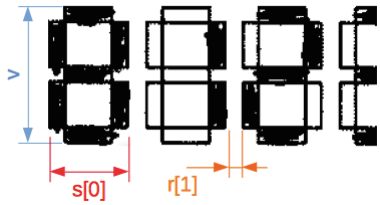
$$s[i] = x\_kraj[i] - x\_pocetak[i] \tag{2.15}$$

a kao prosječna vrijednost širine znamenke uzima se medijan od s.

Računanje visine je najjednostavnije:

$$v = y\_kraj - y\_pocetak \tag{2.16}$$

gdje su $y\_pocetak$ i y_kraj gornja i donja y koordinata niza znamenki određene iz horizontalne projekcije.



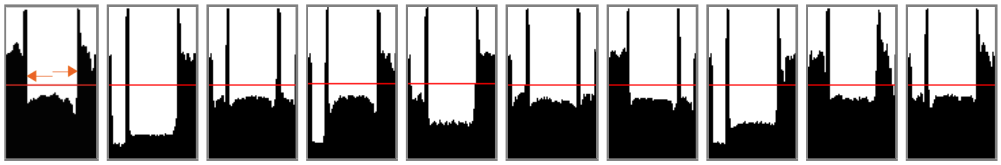
Slika 2.43: Označeni visina, širina i razmak

2.2.4. Širina segmenta znamenke

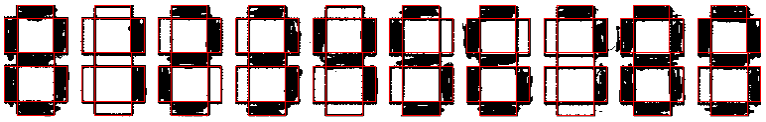
Širina segmenta znamenke računa se iz vertikalne projekcije. Promatra se vertikalna projekcija svake znamenke zasebno. Na 60% visine znamenke postavlja se referentna linija (crvena linija na slikama dolje). Iz sredine te linije kreće se ulijevo i traži kraj lijevog vrha projekcije. Prvi bijeli (crni na slici) slikovni element koji je pronađen predstavlja kraj tog vrha i njegov indeks definira širinu segmenta lijeve strane znamenke. Nakon toga se traži prvi bijeli slikovni element iz sredine prema desnom vrhu projekcije. Kad je pronađen, oduzme se indeks zadnjeg elementa projekcije od tog indeksa i dobiva se širina desnog vrha projekcije.

Po dvije potencijalne širine segmenata svake znamenke spremaju se u polje te se nad elementima tog polja računa aritmetička sredina kako bi se odredila konačna širina segmenta znamenke.

Ignoriraju se preuske ili preširoke širine segmenta, tj. one se ne uzimaju u obzir kod računanja srednje širine segmenta. Za donju granicu postavljena je sedmina širine znamenke, a za gornju trećina širine znamenke.



Slika 2.44: Računanje širine segmenta znamenke

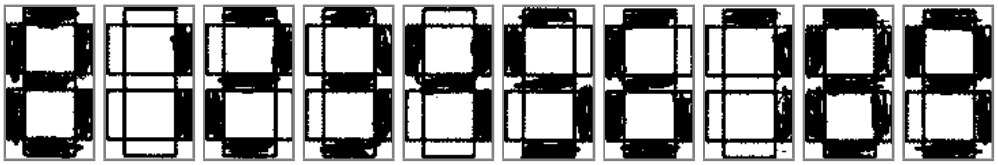


Slika 2.45: Označeni segmenti

2.3. Raspoznavanje

2.3.1. Razrezivanje slike znamenki

Na binarnoj slici se za svaku znamenku određuje regija interesa koja obuhvaća pojedinu znamenku. Regija interesa je pravokutnik kojemu je gornji lijevi rub koordinata $(x\_pocetak[i], y\_pocetak)$, širina $x\_kraj[i] - x\_pocetak[i]$ i visina izračunata prema (2.16). Slikovni elementi označene regije interesa kopiraju se u odredišnu sliku te znamenke. Razrezana ulazna slika:

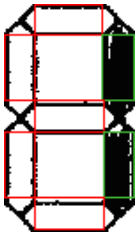


Slika 2.46: Razrezane znamenke

2.3.2. Određivanje granice popunjenosti segmenata

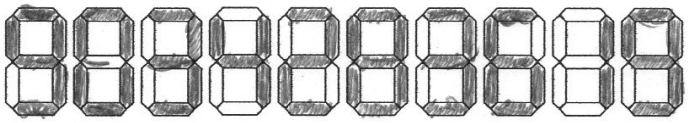
Problemi fiksne granice popunjenosti

Sustav za prepoznavanje JMBAG-ova sa zacrnjenih sedamsegmentnih znamenaka prije je koristio fiksnu granicu popunjenosti segmenta. Za svaki segment znamenke, izračunat je postotak zacrnjenih slikovnih elemenata, te ako je taj postotak manji od 47%, segment je prazan, dok je u suprotnom pun. Na slici 2.47 je prikazana znamenka za koju je ovakav sustav točno raspoznao znamenku. Segmenti koji su označeni kao prazni označeni su crvenom, a puni segmenti označeni su zelenom bojom. Takav način



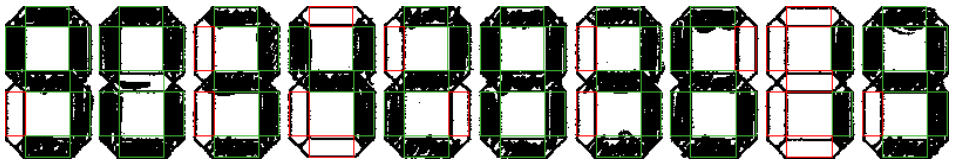
Slika 2.47: Točno raspoznata znamenka

dobar je u većini ispitanih slučajeva, ali se pokazao kao nedovoljno dobar za stvarnu primjenu sustava. Problemi se javljaju kad postoje različiti predlošci znamenaka, s tanjim ili debljim crtama. Predložak s debljim crtama u početku ima veći postotak crnih slikovnih elemenata. Ako bi korisnik nepotpuno popunio predložak s tankim crtama, ali dovoljno da je čovjeku lako iščitati znamenke, moglo bi se dogoditi da pojedini segmenti ne budu prepoznati kao puni zbog praga od 47%. S druge strane, ako bi korisnik popunjavao predložak s debljim crtama i slučajno malo zacrnio pogrešni segment, dok su ostali zacrnjeni segmenti jače ispunjeni, sustav bi taj segment prepoznao kao puni. Slična stvar može se dogoditi i ako pisač malo zakaže pa pojedine segmente ispiše deblje nego što bi trebao. Također, može se dogoditi da se crte malo podebljaju i nakon obrade slike binarizacijom. Na sljedećoj slici prikazan je primjer kad je sustav pogrešno raspoznao znamenku:



Slika 2.48: Originalna slika

Druga znamenka je pogrešno raspoznata zbog srednjeg segmenta koji je pogrešno prepoznat kao puni. Na originalnoj slici je, s obzirom na ostale znamenke, jasno vidljivo da se radi o znamenci 0 te da je segment trebao biti prepoznat kao prazan. Međutim, nakon obrade slike područje tog segmenta je, kako je označeno na drugoj znamenci dolje slijeva, sadržavalo više ili jednako 47% zacrnjenih slikovnih elemenata.



Slika 2.49: Znamenke s označenim punim (zeleno) i praznim (crveno) segmentima

Dinamičko računanje granice popunjenosti

Novi način određivanja granice popunjenosti segmenata određuje prag popunjenosti u toku izvođenja s obzirom na popunjenosti svih segmenata . Prvo se izračunaju postoci popunjenosti svih segmenata cijelog niza znamenaka te se od toga napravi histogram. Histogram je predstavljen kao polje od 101 elementa čije vrijednosti predstavljaju broj segmenata koji imaju postotak popunjenosti jednak odgovarajućem indeksu (postoci

su u rasponu 0 - 100). Na primjeru jedne znamenke, ako imamo segmente sa sljedećim postocima popunjenosti:

$$segment\_0 : 42$$

$$segment\_1 : 75$$

$$segment\_2 : 75$$

$$segment\_3 : 75$$

$$segment\_4 : 21$$

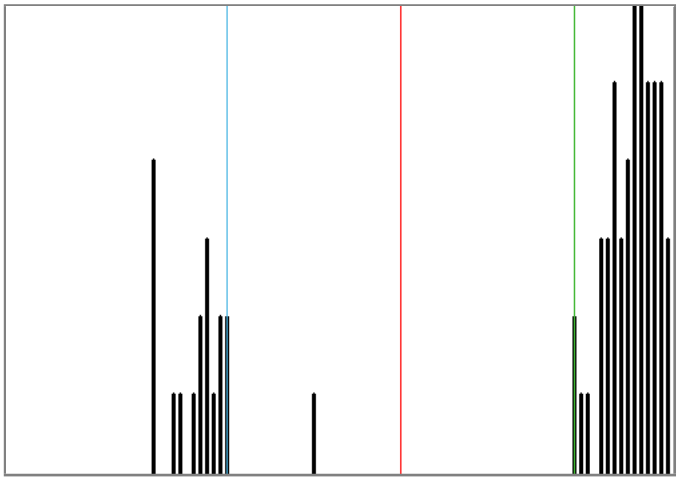
$$segment\_5 : 75$$

$$segment\_6 : 23$$

pripadajuće polje histograma H izgledati će:

$$H[21] = 1$$
$$H[23] = 1$$
$$H[42] = 1$$
$$H[75] = 4$$
$$H[i] = 0,\ za\ i \in [0, 99] \setminus \{21, 23, 42, 75\}.$$

Histogram za sliku 2.48 je prikazan:

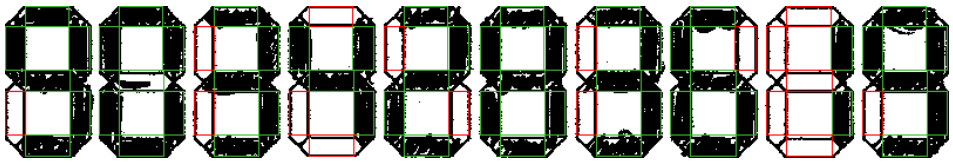


Slika 2.50: Histogram segmenata znamenki

Lijevi brijeg histograma predstavlja prazne segmenta, a desni pune. Kao nova granica popunjenosti uzima se sredina između zadnjeg lijevog praznog segmenta (plavi pretinac na slici), i prvog desnog punog segmenta (zeleni pretinac na slici) te je na slici 2.50 dobivena granica označena crvenom bojom. Prethodno pogrešno raspoznat

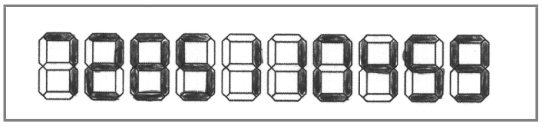
srednji segment druge znamenke na slici je predstavljen usamljenim pretincom između crvenog i plavog pretinca.

S novom granicom od 59%, sve znamenke sa slike 2.48 su točno raspoznate:

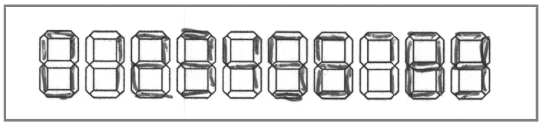


Slika 2.51: Točno raspoznate znamenke

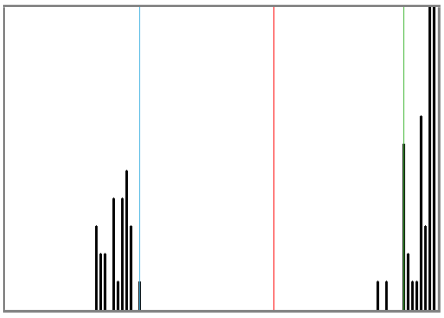
Ovakav način računanja granice otporan je na različite predloške znamenaka i način popunjavanja korisnika (slabije ili jače, ali da je dosljedno) jer su prazni segmenti jednog niza znamenaka slični jedni drugima, kao što su i puni. Histogram takvog niza znamenaka imat će dva glavna brijega koji predstavljaju prazne, odnosno pune segmente, samo će njihov položaj malo varirati od slike do slike.



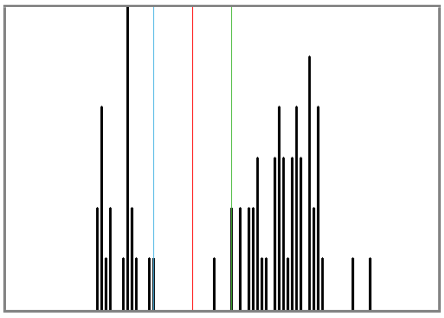
Slika 2.52: Originalna slika 1



Slika 2.54: Originalna slika 2



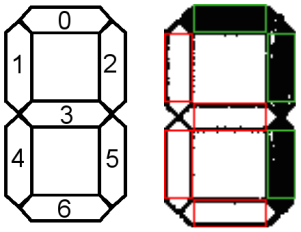
Slika 2.53: Histogram segmenata slike 1



Slika 2.55: Histogram segmenata slike 2

2.3.3. Raspoznavanje znamenaka

Raspoznavanje znamenke zadnji je korak u ovom postupku. Na slici znamenke određuju se regije interesa koje pokrivaju svih sedam segmenata znamenke. Za svaki segment provjerava se postotak ispunjenosti, tj. broj slikovnih elemenata objekta u odnosu na broj svih slikovnih elemenata. Segment je "puni" ukoliko je njegov postotak ispunjenosti jednak ili veći od praga popunjenosti izračunatog kao što je opisano u 2.3.2.



Slika 2.56: Znamenka s brojevima segmenata (lijevo), broj 7(desno)

Na slici desno puni segmenti su uokvireni zelenom, a prazni crvenom bojom.

Svaki segment ima svoju težinu po uzoru na binarni sustav, kao što je navedeno u tablici:

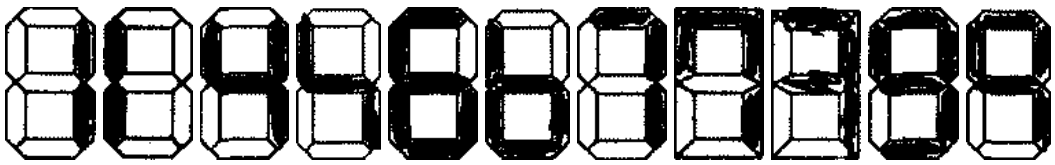
segment	0	1	2	3	4	5	6
težina	1	2	4	8	16	32	64

Tablica 2.1: Segmenti i njihove težine

Zbroj težina punih segmenata jedinstveno određuje neku znamenku. Međutim, jednu znamenku mogu određivati i do tri različita zbroja, ovisno o načinu zapisa pojedinih znamenki (npr. 1, 4, 6, 7, 9). Tablica 2.2 prikazuje znamenke i pripadajuće identifikatore.

znamenka	0	1	2	3	4	5	6	7	8	9
broj(evi)	119	18, 36	93	109	42, 46	107	122, 123	37, 39, 45	127	47, 111

Tablica 2.2: Znamenke s identifikatorima



Slika 2.57: Slike višeznačnih znamenaka

2.3.4. Mjera pouzdanosti raspoznatog niza znamenki

Kako bi se moglo znati s kojom sigurnošću opisani postupak točno raspoznaje JMBAG s ulazne slike, potrebno je uvesti nekakav oblik mjere pouzdanosti. Uvedena mjera pouzdanosti računa se na razini segmenta znamenke na temelju popunjenosti

trenutnog segmenta. Ako se udaljenost popunjenosti segmenta i od praga popunjenosti $P_{popunjenosti}$ definira kao:

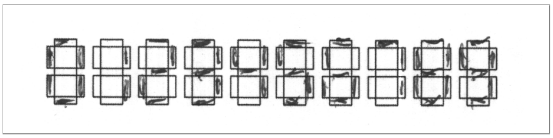
$$r[i] = |P_{popunjenosti} - p[i]|, \tag{2.17}$$

tada se pouzdanost segmenta $p_s[i]$ računa prema:

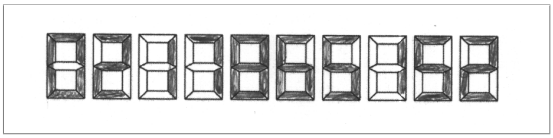
$$p_s[i] = \begin{cases} 50 \cdot \frac{r[i]}{P_{popunjenosti}} + 50 & \text{za } r[i] > P_{pouzdanosti}, p[i] < P_{popunjenosti}, \\ 50 \cdot \frac{r[i]}{100 - P_{popunjenosti}} + 50 & \text{za } r[i] > P_{pouzdanosti}, p[i] \geq P_{popunjenosti}, \\ 50 \cdot \frac{r[i]}{P_{pouzdanosti}} & \text{za } r[i] \leq P_{pouzdanosti}, \end{cases} \tag{2.18}$$

gdje je $P_{pouzdanosti}$ prag koji definira pouzdanost raspoznavanja segmenta kao punog ili praznog, vrijednost mu je postavljena na 15. Pouzdanost će biti u intervalu od 0 do 50 za nepouzdano raspoznate segmente, a u intervalu od 50 do 100 za pouzdano raspoznate segmente.

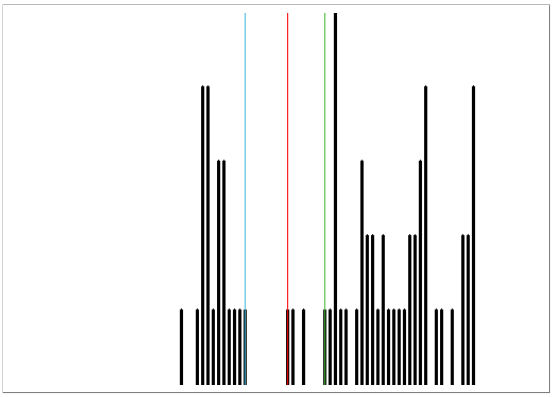
Kao pouzdanost raspoznavanja niza znamenki uzima se najmanji iznos pouzdanosti segmenta, jer ako samo jedan segment neke znamenke nije točno raspoznat, raspoznat će se pogrešna znamenka, a time i pogrešan JMBAG, pa bi se takvu ulaznu sliku trebalo dodatno provjeriti.



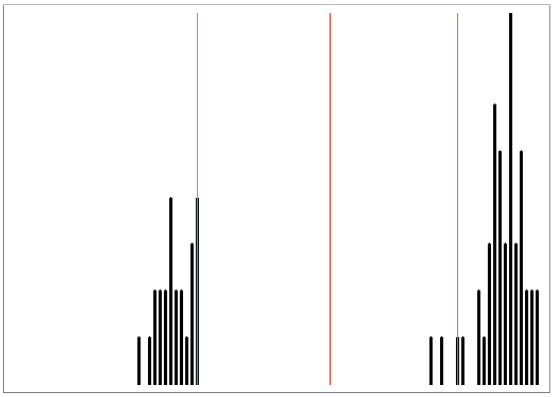
Slika 2.58: Originalna slika 1



Slika 2.60: Originalna slika 2



Slika 2.59: Histogram segmenata slike 1



Slika 2.61: Histogram segmenata slike 2

Predložak na slici 2.58 nije baš dosljedno ispunjen, što se vidi i na histogramu popunjenosti segmenata na slici 2.59, a pouzdanosti najlošijeg segmenta svake znamenke redom iznose:

1. $p_s[0] = 30.8709$
2. $p_s[3] = 28.5886$
3. $p_s[0] = 24.3275$
4. $p_s[2] = 33.0159$
5. $p_s[4] = 23.9506$
6. $p_s[4] = 44.321$
7. $p_s[3] = 5.04504$
8. $p_s[2] = 29.1358$
9. $p_s[0] = 1.44145$
10. $p_s[0] = 11.6516$

Pouzdanost niza sa slike 2.58 je pouzdanost nultog segmenta devete znamenke: $p_s[0] = 1.44145$. S druge strane, predložak na slici 2.60 je ispunjen uredno i dosljedno, a pouzdanosti najlošijeg segmenta svake znamenke redom iznose:

1. $p_s[3] = 74.6251$
2. $p_s[1] = 71.0478$
3. $p_s[4] = 70.4397$
4. $p_s[6] = 70.4441$
5. $p_s[1] = 77.4816$
6. $p_s[4] = 73.9305$
7. $p_s[2] = 74.4179$
8. $p_s[1] = 71.201$
9. $p_s[1] = 81.3879$
10. $p_s[5] = 70.4397$

Pouzdanost niza sa slike 2.60 je 70.4397, što pokazuje i histogram popunjenosti segmenata 2.61.

3. Ispitni skup izvornih slika

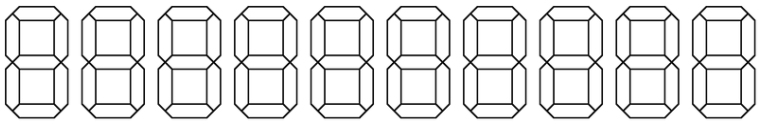
Skup izvornih slika na kojem je provedeno testiranje postupka sadrži 939 slika nizova znamenki skeniranih u 300 dpi s 256 nijansi sive boje. Svaka ulazna slika sastoji se od deset znamenki.

3.1. Predlošci

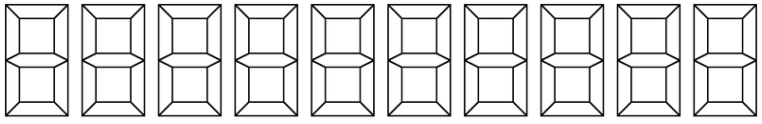
Svaka ulazna slika pripada jednom od četiri tipa predloška:

- predložak 1: 188 primjerka
- predložak 2: 282 primjerka
- predložak 3: 188 primjerka
- predložak 4: 281 primjerak

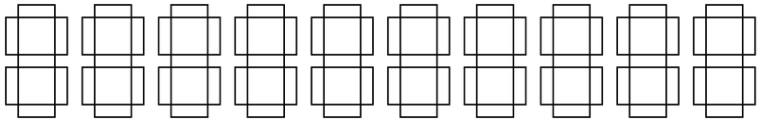
Svaki predložak sadrži deset jednakih uspravnih znamenki s jednakim razmacima između. Okviri segmenata iscrtani su punim linijama crne boje.



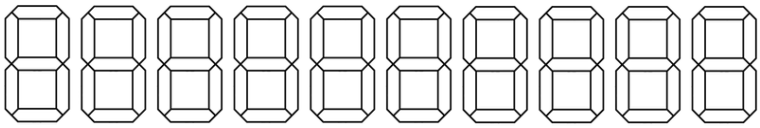
Slika 3.1: Predložak 1



Slika 3.2: Predložak 2



Slika 3.3: Predložak 3



Slika 3.4: Predložak 4

3.2. Podjela po kvaliteti popunjavanja

Skup slika podijeljen je u dva podskupa s obzirom na kvalitetu i urednost popunjavanja predloška. Skup dobrih slika sadrži sve slike za koje bi postupak trebao točno raspoznati niz znamenki s ulazne slike, a skup loših slika sadrži slike za koje je u redu ako postupak netočno raspozna jednu ili više znamenaka. Slike su podijeljene u dva skupa subjektivno.

U skup dobrih slika svrstano je 898 ulaznih slika, a u skup loših 41 slika. Podjela po predlošcima izgleda ovako:

	dobre	loše	ukupno
predložak 1	184 (97.87%)	4(2.13%)	188
predložak 2	270 (95.74%)	12(4.26%)	282
predložak 3	177 (94,15%)	11(5.85%)	188
predložak 4	267 (95.02%)	14(4,98%)	281

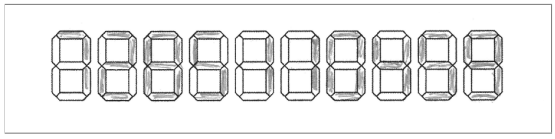
Tablica 3.1: Podjela predložaka po skupovima dobrih i loših slika

3.2.1. Skup dobrih slika

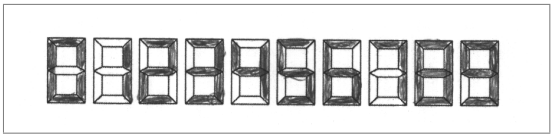
Da bi slika pripadala skupu dobrih slika, segmenti moraju biti dosljedno popunjavani (istom olovkom ili kemijskom olovkom i jednako gusto). Segmenti moraju biti popunjavani unutar okvira.

Postupak je otporan na malo šaranja izvan okvira segmenta i okvira niza znamenki pa to ne utječe na podjelu slika po kvaliteti popunjavanja.

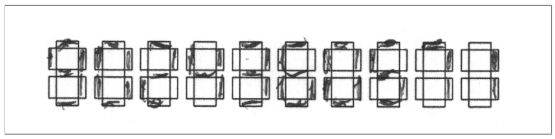
Slijedi nekoliko različitih primjera slika iz ovog skupa s rezultatima postupka.



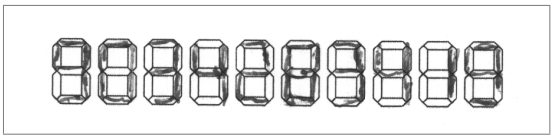
Slika 3.5: Dobra slika 1: 7285110459



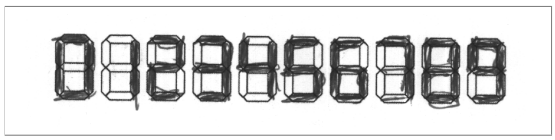
Slika 3.9: Dobra slika 5: 0123456789



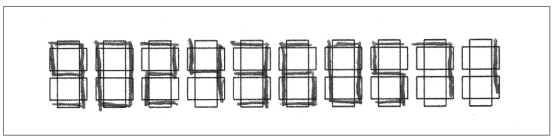
Slika 3.6: Dobra slika 2: 9024360571



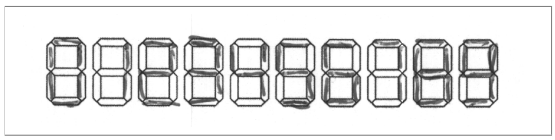
Slika 3.10: Dobra slika 6: 9034283419



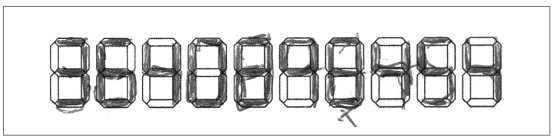
Slika 3.7: Dobra slika 3: 0123456789



Slika 3.11: Dobra slika 7: 9024360571



Slika 3.8: Dobra slika 4: 0123456789

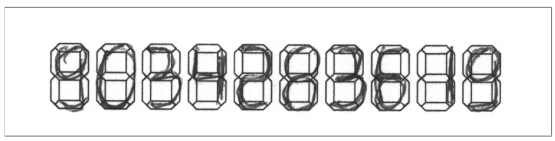


Slika 3.12: Dobra slika 8: 3640898454

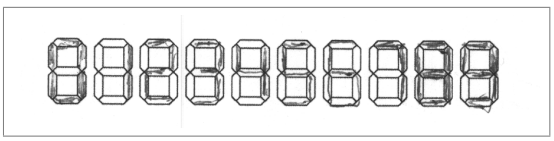
3.2.2. Skup loših slika

Slika pripada skupu loših slika ako je dio segmenata popunjen tamnom bojom, a dio svijetlom. Ako tamno popunjenih segmenata ima više nego svijetlih, prvi prag vršne metode neće biti dovoljno visok da uspješno segmentira svijetle znamenke. Popunjavanje segmenta izvan njegovog okvira ili podebljavanje jednog ruba segmenta neće rezultirati raspoznavanjem segmenta kao punog.

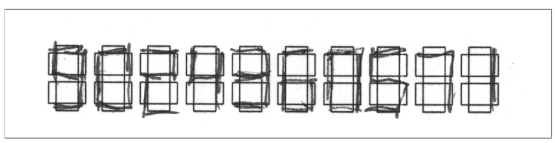
Slijedi nekoliko primjera slika iz lošeg skupa s rezultatima postupka.



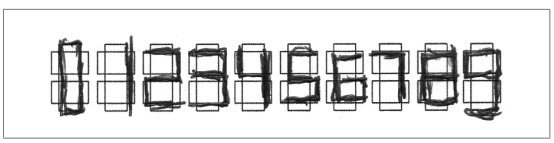
Slika 3.13: Loša slika 1: 90-4263619



Slika 3.15: Loša slika 3: -127—789



Slika 3.14: Loša slika 2: 90-4360511



Slika 3.16: Loša slika 4: 31234–7-9

4. Programska izvedba i vanjske biblioteke

Programska izvedba opisanog postupka ostvarena je u programskom jeziku C++. Korištena je vanjska biblioteka OpenCV[3] koja sadrži raznovrsne strukture podataka i funkcije računalnog vida za rad u stvarnom vremenu.

4.1. Struktura

Struktura programskog koda po datotekama abecednim redom:

- **bht_binary.cc** - konkretan razred koji implementira `binary.cc` s konkretnom implementacijom metode za računanje praga binarizacije BHT algoritmom
- **bht_binary.h**
- **binary.cc** - apstraktni razred s konkretnom metodom za binarizaciju slike i apstraktnom metodom za računanje praga binarizacije (oblikovni obrazac strategija)
- **binary.h**
- **digit.cc** - razred znamenke s metodama za računanje popunjenosti segmenata, histograma segmenata znamenke, računanje širine segmenta za vlastitu znamenku, raspoznavanje vlastite znamenke, računanje pouzdanosti segmenata vlastite znamenke
- **digit.h**
- **digit_helper.cc** - razred sa statičkim pomoćnim metodama za računanje projekcija, histograma i slično
- **digit_helper.h**

- **digit_sequence.cc** - razred niza znamenki koji enkapsulira pretprocesiranje ulazne slike, računanje geometrijskih parametara, raspoznavanje niza znamenki i računanje pouzdanosti raspoznatog niza znamenki
- **digit_sequence.h**
- **drawing_helper.cc** - razred sa statičkim metodama za crtanje projekcija, histograma, vertikalnih ili horizontalnih linija, pravokutnika
- **drawing_helper.h**
- **flags.h** - zaglavlje s eksternim globalnim varijablama za pristup opcijama podešenim kroz parametre naredbenog retka u ostalim datotekama po potrebi
- **general_helper.cc** - razred s raznim statičkim pomoćnim metodama
- **general_helper.h**
- **main.cc** - glavni razred u kojem se pokreće cijeli postupak s odabranom metodom binarizacije
- **mode_method_binary.cc** - konkretan razred koji implementira `binary.cc` s konkretnom implementacijom metode za računanje praga binarizacije vršnom metodom
- **mode_method_binary.h**

4.2. Kratak opis glavnih razreda

4.2.1. ModeMethodBinary

Razred `ModeMethodBinary` enkapsulira kod koji pokušava naći dobru vrijednost za prag binarizacije objašnjenom vršnom metodom (2.1.1). Histogram slike najprije se zaglađuje metodom `smooth_histogram`.

```
int* ModeMethodBinary::smooth_histogram(int *histogram, int size,
    float *kernel, int radius) {

  int *smooth_histogramm = new int[size];
  int kernel_size = 2 * radius + 1;
  double value = 0;

  for (int i = 0; i < size; ++i) {
    value = 0;
    for (int k = 0; k < kernel_size; ++k) {
      if ((i + k - radius) >= 0 && (i + k - radius) < size) {
        value += kernel[k] * histogram[i + k - radius];
      }
```

```
    }
    smooth_histogramm[i] = (int)round(value);
  }
  return smooth_histogramm;
}
```

Metoda `calculate_local_extrems` traži lokalne ekstreme. Metoda za svaki pretinac traži prvi pretinac koji ima različitu vrijednost s njegove lijeve strane i prvi pretinac koji ima različitu vrijednost s njegove desne strane. Kad pronađe oba pretinca, razlikuje tri slučaja. Ako oba pronađena pretinca imaju veću vrijednost od trenutnog pretinca, on je minimum. Ako oba imaju manju vrijednost, pronađen je maksimum. Inače, trenutni pretinac nije ni minimum ni maksimum.

```
void ModeMethodBinary::calculate_local_extrems(int *histogram, int
    size) {
  for (int i = 0; i < size; ++i) {
    int left = 0, right = 0;
    for( left = i - 1; left >= 0 && histogram[i] == histogram[left];
        --left);
    for( right = i + 1; right < size && histogram[i] ==
        histogram[right]; ++right);

    if ((left == -1) && (right == size)) {
      // all histogram values are equal
      break;
    } else if (left == -1) {
      if (histogram[i] < histogram[right]) {
        mins.push_back(right/2);
      } else if (histogram[i] > histogram[right]) {
        maxs.push_back(right/2);
      }
      i = right - 1;
    } else if (right == size) {
      if (histogram[i] < histogram[left]) {
        mins.push_back(ceil((i + left) / 2.0f));
      } else if (histogram[i] > histogram[left]) {
        maxs.push_back(ceil((i + left) / 2.0f));
      }
      break;

    } else {
      if (histogram[left] > histogram[i] && histogram[i] <
          histogram[right]) {
        mins.push_back((left + right) / 2);
        i = right - 1;
      } else if (histogram[left] < histogram[i] && histogram[i] >
          histogram[right]) {
        maxs.push_back((left + right) / 2);
        i = right - 1;
      } else {
        continue;
      }
  }}}
```

4.2.2. Digit

Glavni podatkovni članovi razreda Digit su slika znamenke i pripadna vertikalna projekcija. Njegove zadaće su računanje širine segmenta iz vertikalne projekcije i računanje histograma popunjenosti segmenata vlastite znamenke te prosljeđivanje rezultata pozivatelju (DigitSequence). Nakon što pozivatelj na temelju histograma segmenata svih znamenaka izračuna granicu popunjenosti segmenta, u razredu Digit se za svaki segment određuje je li pun ili prazan, a zatim se na temelju punih segmenata znamenke računa jedinstveni binarni zbroj koji jednoznačno određuje znamenku. U razredu Digit računa se pouzdanost svakog segmenta vlastite znamenke te se kao rezultat pozivatelju vraća najmanja pouzdanost.

Metoda `calculate_binary_segments_sum` računa sumu binarnih oznaka segmenata.

```
int Digit::calculate_binary_segments_sum(float threshold, bool
    *segment_fullness) {

  int sum = 0;

  for (int i = 0; i < 7; ++i) {
    if (is_full(i, threshold)) {
      sum += pow(2, i);
      segment_fullness[i] = 1;
    }
  }
  return sum;
}
```

Metoda `get_digit_value` prepoznaje o kojem je zacrnjenom broju riječ, u ovisnosti o izračunatoj binarnoj sumi.

```
char Digit::get_digit_value() {
  char digit_value;
  switch(binary_sum) {
    case 119:
      digit_value = '0';
      break;
    case 18:
    case 36:
      digit_value = '1';
      break;
    case 93:
      digit_value = '2';
      break;
    case 109:
      digit_value = '3';
      break;
    case 42:
    case 46:
      digit_value = '4';
```

```
    break;
  case 107:
    digit_value = '5';
    break;
  case 122:
  case 123:
    digit_value = '6';
    break;
  case 37:
  case 39:
  case 45:
    digit_value = '7';
    break;
  case 127:
    digit_value = '8';
    break;
  case 47:
  case 111:
    digit_value = '9';
    break;
  default:
    digit_value = '-';
    break;
 }
 return digit_value;
}
```

Pouzdanost trenutne znamenke računa se pomoću metode `calculate_reliability`.

```
#define RELIABILITY_SPAN 15.0f

float Digit::calculate_reliability(float threshold) {
 float difference = 100.0f;
 float fullness = 0.0f;
 for (int i = 0; i < 7; ++i) {
  float segment_reliability = fabs(threshold -
     segment_fill_percentage[i]);
  if (segment_reliability < difference) {
    difference = segment_reliability;
    fullness = segment_fill_percentage[i];
  }
 }
 if (difference > RELIABILITY_SPAN) {
  if (fullness < threshold) {
    return 50 + 50 * difference / (threshold);
  } else {
    return 50 + 50 * difference / (100.0f - threshold);
  }
 } else {
  return difference * (10 / 3.0f) ;
 }}
```

4.2.3. DigitSequence

Razred DigitSequence obavlja pretprocesiranje slike pomoću metode `preprocess`.

```
void DigitSequence::preprocess() {
  cv::Mat img_original_copy = img_original.clone();
  histogram = DigitHelper::get_histogram(img_original_copy);
  int threshold = binary->get_threshold(histogram, 0, 256);

  binary->binarize(img_original_copy, threshold, &img_binary);
  center_point = calculate_center_point(img_binary);
  angle = calculate_angle(img_binary);
}
```

Nad svakim objektom razreda Digit poziva metodu koja prepoznaje zacrnjen broj korištenjem opisanih metoda.

```
void DigitSequence::get_jmbag() {
  for (int i = 0; i < DIGIT_COUNT; ++i) {
    std::cout << digits[i]->recognize_digit(threshold,
        segment_fullness[i]);
  }
}
```

Računanje pouzdanosti prepoznavanja niza zacrnjenih brojeva oslanja se na izračune koje obavljaju objekti razreda Digit.

```
float DigitSequence::calculate_reliability() {

  if (fullness_threshold < 20.0f) {
    return -1.0f;
  }

  float minimum_reliability = 100.0f;
  for (int i = 0; i < DIGIT_COUNT; ++i) {
    float digit_reliability =
        digits[i]->calculate_reliability(fullness_threshold);
    if (digit_reliability < minimum_reliability) {
      minimum_reliability = digit_reliability;
    }

  }
  reliability = minimum_reliability;
  return reliability;
}
```

4.3. Koraci postupka

Glavni i pojednostavljeni koraci postupka raspoznavanja JMBAG-a s ulazne slike:

1. Učitavanje slike i postavljanje parametara (broj znamenki, postotak širine znamenki) i zastavica (ispis varijabli, prikaz slika i slično), ako su predani preko argumenata naredbenog retka.
2. Inicijalizacija DigitSequence objekta s brojem znamenaka i metodom binarizacije (vršna metoda).
3. Pretprocesiranje ulazne slike.
   (a) računanje pragova binarizacije pomoću vršne metode
   (b) binarizacija ulazne slike s postavljenim pragom
   (c) računanje orijentacije slike
   (d) rotiranje slike ako je apsolutna vrijednost kuta glavne osi objekta veća od 0.05 °
4. Računanje svojstava i procjena parametara
   (a) računanje horizontalne i vertikalne projekcije binarne slike
   (b) računanje rubnih točaka znamenaka - ako se ne mogu izračunati rubne točke zbog preniskog praga binarizacije, postavlja se drugi prag vršne metode i kreće na korak 3. a)
   (c) podjela slike niza znamenaka na N odvojenih slika znamenaka
   (d) računanje širine segmenta iz vertikalne projekcije
   (e) računanje histograma popunjenosti svih segmenata
   (f) računanje praga popunjenosti iz histograma segmenata - ako je prag popunjenosti manji od 20%, postavlja se drugi prag vršne metode i kreće na korak 3. a)
5. Raspoznavanje znamenaka
   (a) uspoređivanje postotka popunjenosti segmenata s pragom popunjenosti
   (b) raspoznavanje znamenke po znamenke na temelju binarnog zbroja punih segmenata za trenutnu znamenku

4.4. Korištenje programa

Izvornom kodu priložen je i *Makefile* za jednostavno prevođenje izvornog koda. Za prevođenje je samo potrebno napisati `make` u konzoli, a nakon prevođenja, program se pokreće s

```
./main <put_do_slike>.
```

Program dohvaća ulaznu sliku niza znamenki preko prvog argumenta naredbenog retka. Program podržava i nekoliko opcija za prikaz dodatnih informacija te par opcija za podešavanje nekih parametara, a popis svih opcija može se dobiti s argumentom `--help`.

Opcije za prikaz dodatnih informacija su:

`--debug` ispis međukoraka i varijabli (prag binarizacije, rubne točke znamenaka, parametri znamenke, popunjenosti segmenata, ...)

`--show` prikazivanje slika međukoraka (projekcije s pragovima, histogrami, slike pojedinačnih znamenaka, ...)

`--save` spremanje slika međukoraka u trenutni direktorij

`--folder <name>` spremanje slika međukoraka u postojeći direktorij zadan ovom opcijom

`--prefix` prefiks za imena slika kod spremanja slika u direktorij

Opcije za podešavanje:

`-N <count>` broj znamenaka na slici, pretpostavljena vrijednost je 10

`-P <percentage>` postotak širine svih znamenaka zajedno na području od početka do kraja niza znamenki, pretpostavljena vrijednost je 0.85 (85% širine niza znamenki zauzimaju znamenke, a 15% razmaci)

4.5. Pristupna komponentna u Javi

U sklopu ovog rada razvijena je i pristupna komponenta koja omogućuje korištenje biblioteke iz programskog jezika Java. Biblioteka nudi jednostavno sučelje za korištenjeglavne funkcionalnosti.

```
public class SevenSegment {

  public native String recognize(String fileName);

  public native String recognize(String fileName, int digitCount);

  public native String recognize(String fileName, float
     totalDigitWidhtPercentage);

  public native String recognize(String fileName, int digitCount,
     float totalDigitWidhtPercentage);

}
```

5. Eksperimentalni rezultati

5.1. Uspješnost

Testiranje je provedeno na skupu od 939 slika koje su podijeljene u dvije grupe, dobre i loše (opisano u poglavlju 3). U daljnjem tekstu slijede rezultati testiranja postupka prije i nakon poboljšanja te rezultati po tipu predloška. Podaci u tablicama pokazuju uspješnost raspoznavanja na razini znamenke i na razini cijelog niza znamenki.

Kod računanja rezultata ignoriraju se nevažeće znamenke. Na primjer, ako je na predlošku ispunjeno 01234567H9, gleda se 9 od 10 znamenaka, pošto 'H' nije važeća znamenka. Ako je svih ostalih 9 znamenaka uspješno raspoznato, gleda se kao da je raspoznat cijeli niz znamenaka. Zbog toga su brojevi 'ukupno' u tablicama na razini znamenaka manji od broja ukupnih nizova pomnoženih s duljinom niza (10).

5.1.1. Uspješnost postupka prije poboljšanja

U tablicama 5.1 i 5.2 prikazani su rezultati testiranja postupka za raspoznavanje matičnog broja na ispitnom skupu slika prije poboljšanja.

Kao što je ranije navedeno, prije poboljšanja postupak je koristio BHT metodu traženja praga za binarizaciju, fiksnu granicu popunjenosti segmenta od 47% te je bio puno osjetljiviji na ispunjavanje segmenata izvan okvira i malo šaranja oko slike niza znamenki zbog načina određivanja rubnih točaka iz projekcija. Ponekad se nije mogla procjeniti širina segmenta na horizontalnoj projekciji, ako referentna linija ne bi sjekla tri glavna brijega pa bi se ovdje postupak izvođenja zaustavio. Kod rotacije slike bila je korištena bilinearna interpolacija zbog koje je bilo potrebno rotirati originalnu sliku te ju ponovo binarizirati. Histogram rotirane slike se za neke ulazne slike dosta razlikovao od slike originalne nerotirane slike, što je u nekim slučajevima rezultiralo uspješnom binarizacijom prije rotacije, i neuspješnom nakon (npr. vidljivi samo okviri segmenata).

	ukupno	raspoznato	neraspoznato	uspješnost
dobre	898	778	120	86.64%
loše	41	3	38	7.32%

Tablica 5.1: Uspješnost prije poboljšanja na razini niza znamenki (na razini slike)

	ukupno	raspoznato	neraspoznato	uspješnost
dobre	8935	8562	373	95.83%
loše	406	254	152	62.56%

Tablica 5.2: Uspješnost prije poboljšanja na razini znamenke

5.1.2. Uspješnost postupka nakon poboljšanja

U tablicama 5.3 i 5.4 prikazani su rezultati testiranja postupka za raspoznavanje matičnog broja na ispitnom skupu slika nakon poboljšanja.

Postupak je poboljšan korištenjem vršne metode za određivanje praga za binarizaciju, korištenjem interpolacijske metode najbližeg susjeda koja dozvoljava rotiranje direktno binarne slike, robusnijim određivanjem rubnih točaka znamenaka, procjenjivanjem širine segmenta iz vertikalne projekcije i dinamičkim računanjem praga popunjenosti segmenta na histogramu segmenta znamenki. Dodano je i automatsko pokretanje cijelog postupka ispočetka s izračunatim drugim pragom vršne metode u slučaju da se parametri znamenaka ne mogu procjeniti ili je granica popunjenosti segmenata niža od 20% (u slučajevima svijetlih ulaznih slika - binarna slika na kojoj su vidljivi samo okviri segmenata).

	ukupno	raspoznato	neraspoznato	uspješnost
dobre	898	887	11	98.78%
loše	39	12	27	30.77%

Tablica 5.3: Uspješnost nakon poboljšanja na razini niza znamenki (na razini slike)

	ukupno	raspoznato	neraspoznato	uspješnost
dobre	8935	8924	11	99.88%
loše	386	333	53	86.27%

Tablica 5.4: Uspješnost nakon poboljšanja na razini znamenke

5.1.3. Uspješnost po predlošcima

Rezultati iz tablica 5.5 i 5.6 pokazuju da su predlošci 1 i 4 uspješniji od predložaka 2 i 3. Većini ispitanika najmanje se svidio predložak 3 te im ga je bilo najteže ispuniti.

	ukupno	raspoznato	neraspoznato	uspješnost
predložak 1	188	185	3	98.40%
predložak 2	282	265	17	93.97%
predložak 3	188	177	11	94.15%
preložak 4	281	272	9	96.79%

Tablica 5.5: Uspješnost po predlošcima na razini niza znamenki

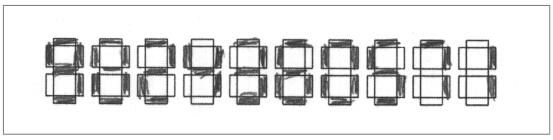
	ukupno	raspoznato	neraspoznato	uspješnost
predložak 1	1867	1859	8	99.57%
predložak 2	2807	2784	23	99.18%
predložak 3	1870	1851	19	98.98%
predložak 4	2797	2780	17	99.39%

Tablica 5.6: Uspješnost po predlošcima na razini znamenke

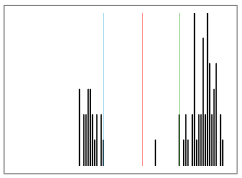
5.2. Analiza rezultata

U idealanom slučaju uspješno je raspoznata svaka znamenka s ulazne slike, tj. program vraća isti niz znamenaka koji se nalazi na slici. Da bi program mogao potpuno točno očitati sve znamenke, slika mora zadovoljiti uvjete opisane u poglavlju 3.2.1. U skupu dobrih slika potpuno je raspoznato 98.78% nizova znamenki.

Slijedi popis preostalih 11 ulaznih slika koje uglavnom nisu točno raspoznate jer je korisnik zabunom ispunio pogrešni segment na pola. Za svaki ulazni niz prikazani su originalna slika i histogram segmenata znamenki s označenim pragom popunjenosti. S desne strane tih slika prikazani su i očekivani izlaz postupka, izlaz postupka, prag popunjenosti segmenta, pouzdanost kao i popunjenost svih segmenata znamenke koja je pogrešno raspoznata za zadani ulazni niz.



Slika 5.1: Dobra slika 1

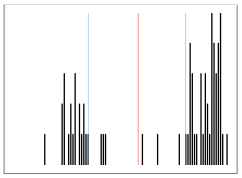


Slika 5.2: Histogram segmenata dobre slike 1

$ocekivani\_izlaz$: 9024360571
$izlaz$: 9824360571
$prag\_popunjenosti$: 60
$pouzdanost$: 21.05

$segment[1][0]$: 77.3684
$segment[1][1]$: 79.619
$segment[1][2]$: 80.9524
$segment[1][3]$: 66.3158
$segment[1][4]$: 91.4815
$segment[1][5]$: 84.0741
$segment[1][6]$: 88.2456

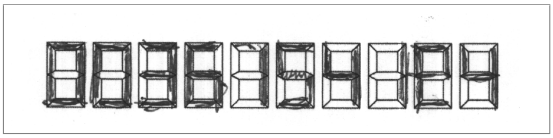
Slika 5.3: Dobra slika 2



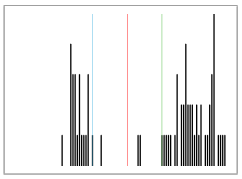
Slika 5.4: Histogram segmenata dobre slike 2

$ocekivani\_izlaz$: 0217865152
$izlaz$: 0−17865152
$prag\_popunjenosti$: 58
$pouzdanost$: 9.87

$segment[1][0]$: 96.7128
$segment[1][1]$: 35.4839
$segment[1][2]$: 89.5636
$segment[1][3]$: 100
$segment[1][4]$: 94.4742
$segment[1][5]$: 60.9626
$segment[1][6]$: 83.218



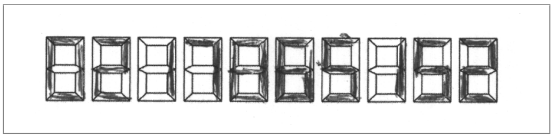
Slika 5.5: Dobra slika 3



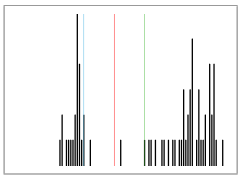
Slika 5.6: Histogram segmenata dobre slike 3

$ocekivani\_izlaz$: 00361541−4
$izlaz$: 00367541−4
$prag\_popunjenosti$: 53
$pouzdanost$: 18.29

$segment[4][0]$: 58.4874
$segment[4][1]$: 31.254
$segment[4][2]$: 97.2426
$segment[4][3]$: 35.2941
$segment[4][4]$: 34.9481
$segment[4][5]$: 91.6955
$segment[4][6]$: 27.8992



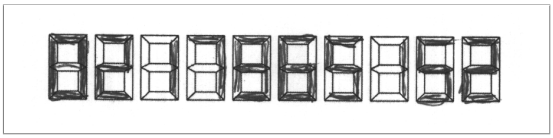
Slika 5.7: Dobra slika 4



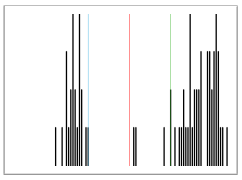
Slika 5.8: Histogram segmenata dobre slike 4

$ocekivani\_izlaz$: 0217865152
$izlaz$: 0217885152
$prag\_popunjenosti$: 47
$pouzdanost$: 10.00

$segment[5][0]$: 91.8403
$segment[5][1]$: 88.8672
$segment[5][2]$: 50
$segment[5][3]$: 75.6944
$segment[5][4]$: 100
$segment[5][5]$: 97.9779
$segment[5][6]$: 92.3611

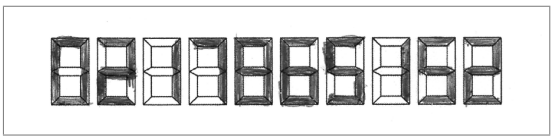


Slika 5.9: Dobra slika 5

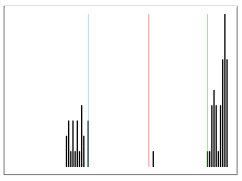


Slika 5.10: Histogram segmenata dobre slike 5

$segment[0][0]$: 91.0473
$segment[0][1]$: 93.5606
$segment[0][2]$: 99.8106
$segment[0][3]$: 57.6014
$segment[0][4]$: 94.3015
$segment[0][5]$: 91.1765
$segment[0][6]$: 97.6351



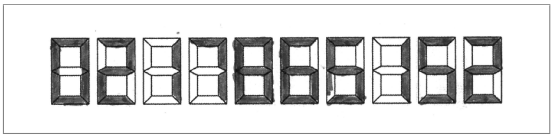
Slika 5.11: Dobra slika 6



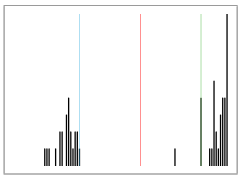
Slika 5.12: Histogram segmenata dobre slike 6

$ocekivani\_izlaz$: 0217865152
$izlaz$: 0−17865152
$prag\_popunjenosti$: 63
$pouzdanost$: 9.61

$segment[1][0]$: 93.4454
$segment[1][1]$: 31.3149
$segment[1][2]$: 92.2145
$segment[1][3]$: 100
$segment[1][4]$: 98.9916
$segment[1][5]$: 65.8824
$segment[1][6]$: 96.9748



Slika 5.13: Dobra slika 7

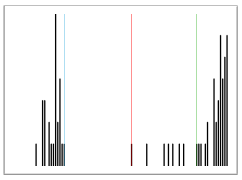


Slika 5.14: Histogram segmenata dobre slike 7

$ocekivani\_izlaz$: 0217865152
$izlaz$: 0217866152
$prag\_popunjenosti$: 59
$pouzdanost$: 69.7239

$segment[6][0]$: 96.5278
$segment[6][1]$: 100
$segment[6][2]$: 22.8571
$segment[6][3]$: 99.8264
$segment[6][4]$: 75.1736
$segment[6][5]$: 98.2639
$segment[6][6]$: 93.2292

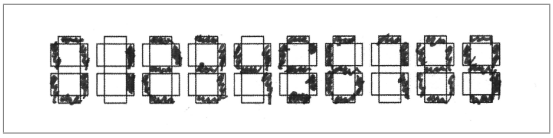
Slika 5.15: Dobra slika 8



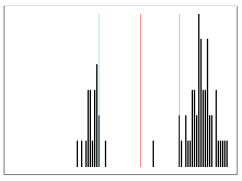
Slika 5.16: Histogram segmenata dobre slike 8

$ocekivani\_izlaz$: 003615412−
$izlaz$: 00361541−−
$prag\_popunjenosti$: 55
$pouzdanost$: 2.42

$segment[8][0]$: 92.7928
$segment[8][1]$: 22.3423
$segment[8][2]$: 97.8378
$segment[8][3]$: 100
$segment[8][4]$: 99.3162
$segment[8][5]$: 55.7265
$segment[8][6]$: 96.9369



Slika 5.17: Dobra slika 9

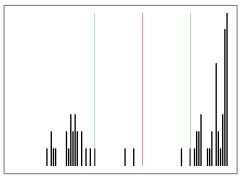


Slika 5.18: Histogram segmenata dobre slike 9

$ocekivani\_izlaz$: 0123456789
$izlaz$: 0123496789
$prag\_popunjenosti$: 59
$pouzdanost$: 22.47

$segment[5][0]$: 92.2523
$segment[5][1]$: 91.8519
$segment[5][2]$: 65.7407
$segment[5][3]$: 89.1892
$segment[5][4]$: 40
$segment[5][5]$: 86.1404
$segment[5][6]$: 99.4595

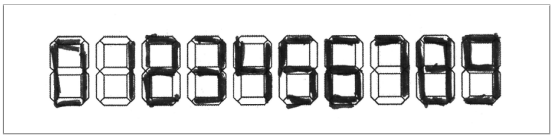
Slika 5.19: Dobra slika 10



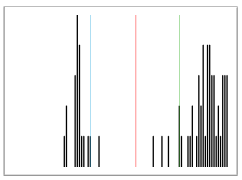
Slika 5.20: Histogram segmenata dobre slike 10

$ocekivani\_izlaz$: 0217865152
$izlaz$: 0217866152
$prag\_popunjenosti$: 60
$pouzdanost$: 5.44

$segment[6][0]$: 90
$segment[6][1]$: 98.3929
$segment[6][2]$: 32.1429
$segment[6][3]$: 99.6875
$segment[6][4]$: 61.6319
$segment[6][5]$: 99.1319
$segment[6][6]$: 98.2812



Slika 5.21: Dobra slika 11



Slika 5.22: Histogram segmenata dobre slike 11

$ocekivani\_izlaz$: 0123456789
$izlaz$: 01−3456789
$prag\_popunjenosti$: 57
$pouzdanost$: 27.54

$segment[2][0]$: 97.6068
$segment[2][1]$: 65.2632
$segment[2][2]$: 88.2456
$segment[2][3]$: 91.6239
$segment[2][4]$: 86.6667
$segment[2][5]$: 40.8333
$segment[2][6]$: 94.7009

5.3. Vrijeme izvođenja

Prosječno vrijeme izvođenja postupka za jednu sliku prije poboljšanja iznosilo je 164 milisekunde, a prosječno vrijeme izvođenja postupka nakon poboljšanja iznosi 29 milisekundi. Mjerenje je izvršeno na ulaznim slikama veličine 900x190 slikovnih elemenata na procesoru Intel Core i5 osnovne frekvencije 1.3 GHz (oznaka i5-4250U).

6. Zaključak

Predstavljena su poboljšanja sustava za raspoznavanje matičnog broja iz sedamsegmentnih znamenaka u odnosu na prvu inačicu postupka. BHT metoda određivanja praga binarizacije zamijenjena je s vršnom metodom koja je pogodnija za primjenu u ovom slučaju na bimodalnim ili trimodalnim histogramima. Vršnom metodom računaju se čak dva praga binarizacije, za slučaj svijetlog ispunjavanja predloška s tamnim okvirima segmenata. Interpolacijska metoda kod rotacije slike je promijenjena s bilinearne interpolacije na interpolaciju najbližeg susjeda te je ovdje uštađeno malo procesorskog vremena zbog direktne rotacije binarne slike. Traženje rubnih točaka na projekcijama uz dva praga osigurava manju osjetljivost na manje ili veće šumove ili mrlje izvan područja segmenata. Procjena širine segmenta iz vertikalnih projekcija znamenaka je jednostavnija i preciznija od procjene iz horizontalne projekcije. I na kraju, dinamičko određivanje granice popunjenosti segmenata iz histograma popunjenosti segmenata pokazalo se kao bolje rješenje od fiksne granice popunjenosti. Fiksna granica može biti previsoka u slučajevima slabijeg popunjavanja segmenata, a preniska u slučaju gustog popunjavanja većine segmenata i, na primjer, jednog vrlo slabo popunjenog segmenta koji je rezultat korisnikove pogreške i ne bi trebao biti raspoznat kao puni.

Testni skup slika je podijeljen po kvaliteti ispunjavanja predložaka na skup dobrih i skup loših slika. U skupu loših slika nalaze se slike za koje se od sustava ne očekuje točno raspoznavanje, a uglavnom su to slike sa zacrnjivanjem izvan okvira segmenata pa stoga ima smisla promatrati uspješnosti takvih grupa slika odvojeno.

Uspješnost raspoznavanja slika iz dobrog skupa se u odnosu na staru inačicu postupka povećala s 86.64% na 98.78% ako se gleda raspoznavanje cijelog niza (cijele slike), odnosno s 95.83% na 99.88% na razini jedne znamenke. Prosječno vrijeme obrade jedne slike smanjeno je sa 164 milisekunde na 29 milisekundi. Ovakvi rezultati djeluju obećavajuće te bi se sustav mogao koristiti na stvarnom problemu.

LITERATURA

[1] Wrapping a c++ library with jni, siječanj 2012. URL http://thebreakfastpost.com/2012/01/23/wrapping-a-c-library-with-jni-part-1/.

[2] Balanced histogram thresholding, listopad 2013. URL http://en.wikipedia.org/wiki/Balanced_histogram_thresholding.

[3] Opencv documentation, travanj 2014. URL http://docs.opencv.org/.

[4] Principal components analysis, lipanj 2014. URL http://en.wikipedia.org/wiki/Principal_component_analysis.

[5] Ines Bonačić, Tomislav Herman, Tomislav Krznar, Edin Mangić, Goran Molnar, i Marko Čupić. *Optical Character Recognition of Seven–segment Display Digits Using Neural Networks*. 2009.

[6] Dražen Dostal. *Detekcija stanične jezgre u mikroskopskim slikama*. Završni rad, FER, 2009.

[7] Melita Kokot. *Strojno očitanje matičnog broja zadanog zacrnjivanjem elemenata sedamsegmentnih znamenki*. Završni rad, 2012.

[8] Robert Laganiere. *OpenCV 2 Computer Vision Application Programming Cookbook*. Packt Publishing, 2011.

[9] Brian G. Schunck Ramesh Jain, Rangachar Kasturi. *Machine Vision*. McGraw-Hill, 1995.

[10] Lindsay I Smith. A tutorial on principal components analysis, veljača 2002.

Izvedba biblioteke za strojno očitanje rukom popunjavanih sedamsegmentnih znamenki

Sažetak

Strojno očitavanje broja iz skeniranog obrasca s poljem sa zacrnjenim poljima sedamsegmentnih znamenaka omogućava automatsku identifikaciju korisnika. Program učitava sivu ulaznu sliku sa sedamsegmentnim znamenkama pa ju binarizira koristeći vršnu metodu za izračun praga. Pomoću analize glavnih komponenata objekta (PCA) računa se glavna os pružanja kako bi se slika rotirala, ukoliko os nije poravnata sa horizontalom. Iz horizontalnih i vertikalnih projekcija binarne slike određene su dimenzije i razmak između znamenaka, širina segmenta te lokacije znamenaka. Binarna slika se dijeli u više malih slika od kojih svaka sadrži jednu znamenku. Pomoću širine znamenke postavljaju se regije interesa na svaki od sedam segmenata znamenke i njihovom analizom moguće je odrediti o kojoj znamenci se radi. Za evaluaciju rješenja korišteno je 939 slika sa sedamsegmentnim znamenkama koje su podijeljene u dvije grupe, ovisno o kvaliteti ispunjavanja. Implementirano je jednostavno sučelje koje omogućava korištenje biblioteke iz programskog jezika Java.

Ključne riječi: bht, binarna slika, binarizacija, vršna metoda, PCA, rotacija slike, vertikalna projekcija, horizontalna projekcija, lokalizacija znamenki, sedamsegmentna znamenka

Implementation of a library for recognizing hand-filled seven-segment digits

Abstract

Machine reading of the scanned form with a region with blackened fields of seven-segment digits allows automatic identification of users. The program loads the input grayscale image of seven-segment digits and then binarizes it using the mode method to calculate the threshold. Using principal components analysis of the object (PCA) it computes the orientation of the major axis to rotate the image, if the axis is not aligned with the horizontal line. The dimensions and spacing between the digits, width of segment and location of the digits are determined by the horizontal and vertical projection of the binary image. The binary image is divided into several small images each containing a single digit. Using the width of a segment, regions of interest are placed on each of the seven-segment digits and their analysis can determine which digit is on image. For evaluation of the solutions, 939 images with seven-segment digits are used. Images are divided into two groups, depending on the quality of compliance. A simple interface is defined so that the library can be used with the Java programming language.

Keywords: bht, binary histogram thresholding, binary image, binarization, mode method, horizontal projection, vertical projection, PCA, principal component analysis, image rotation, localization of digits, seven-segment digit