UNIVERZA V MARIBORU

FAKULTETA ZA ELEKTROTEHNIKO,
RAČUNALNIŠTVO IN INFORMATIKO

Marko Kočevar

PREPOZNAVANJE BESEDILA V DIGITALNEM VIDEO SIGNALU

Diplomsko delo

MARIBOR, september 2010

Diplomsko delo univerzitetnega študijskega programa

PREPOZNAVANJE BESEDILA V DIGITALNEM VIDEO SIGNALU

Študent:	Marko Kočevar
Študijski program:	Univerzitetni, Elektrotehnika
Smer:	elektronika
Mentor(ica):	izr. prof. dr. Matej Rojc
Somentor(ica):	red. prof. dr. Zdravko Kačič

Maribor, september 2010

FERI
Fakulteta za elektrotehniko, računalništvo in informatiko

Smetanova ulica 17
2000 Maribor

Številka: E-2626
Datum in kraj: 17. 09. 2010, Maribor

Na osnovi 330. člena Statuta Univerze v Mariboru (Ur. l. RS, št. 1/2010)

SKLEP O DIPLOMSKEM DELU

1. **Marku Kočevarju,** študentu univerzitetnega študijskega programa ELEKTROTEHNIKA, smer Elektronika, se dovoljuje izdelati diplomsko delo pri predmetu Teorija vezij.

2. **MENTOR: izred. prof. dr. Matej Rojc**
   SOMENTOR: red. prof. dr. Zdravko Kačič

3. **Naslov diplomskega dela:**
   PREPOZNAVANJE BESEDILA V DIGITALNEM VIDEO SIGNALU

4. **Naslov diplomskega dela v angleškem jeziku:**
   RECOGNITION OF TEXT IN DIGITAL VIDEO SIGNAL

5. Diplomsko delo je potrebno izdelati skladno z "Navodili za izdelavo diplomskega dela" in ga oddati v treh izvodih (en vezan izvod in dva nevezana izvoda) ter en izvod elektronske verzije do 17. 09. 2011 v referatu za študentske zadeve.

Pravni pouk: Zoper ta sklep je možna pritožba na senat članice v roku 3 delovnih dni.

Dekan:

UNIVERZA V MARIBORU
FAKULTETA ZA ELEKTROTEHNIKO, RAČUNALNIŠTVO IN INFORMATIKO
8

Obvestiti:
- kandidata,
- mentorja,
- somentorja,
- odložiti v arhiv.

ZAHVALA

Iskreno se zahvaljujem mentorju Izr. prof. dr. Mateju Rojcu in red. prof. dr. Zdravku Kačiču za vsestransko pomoč pri izdelavi diplomske naloge.

Posebno se zahvaljujem tudi mojim staršem in vsem ostalim, ki so me podpirali pri študiju.

PREPOZNAVANJE BESEDILA V DIGITALNEM VIDEO SIGNALU

Ključne besede: procesiranje slike, zaznavanje in lokalizacija besedila, segmentacija in binarizacija, Sobel operator, OCR.

UDK: 004.932.75 (043.2)

Povzetek

V diplomskem delu smo predstavili postopek in izvedbo prepoznavanje besedila v digitalnem video signalu. Natančneje smo predstavili vse korake, ki so potrebni v fazi prepoznavanje besedila. Ti koraki so: zaznavanje in lokalizacija besedila, segmentacija in binarizacija besedila in na koncu še optično prepoznavanje znakov (OCR). V diplomskem delu smo opisali strojno in tudi programsko opremo, ki smo ju uporabili za prepoznavanje besedila v digitalnem video signalu.

RECOGNITION OF TEXT IN DIGITAL VIDEO SIGNAL

Key words: image processing, text detection and localization, segmentation and binarization, Sobel operator, OCR.

UDK: 004.932.75 (043.2)

Abstract

In this diploma thesis we have described the process and realization of text recognition in digital video signal. We have described all individual steps, which are needed in order to process the text in the digital video signal. These steps are: the text recognition and localization, segmentation and binarisation, and in the end also optical character recognition (OCR). In this diploma thesis we have described also hardware and software part, which were used for text recognition process in digital video signal.

VSEBINA

1 UVOD **1**

1.1 Ideja diplomskega dela ... 1

1.2 Struktura diplomskega dela ... 1

2 ZASNOVA SITEMA **3**

3 BESEDILO V SLIKI IN VIDEO SIGNALU **5**

4 PROCES IZLOČANJA BESEDILA **7**

4.1 Opis iskanja štirirobnih objektov ... 8

4.1.1 Spreminjanje velikosti slike ... 8

4.1.2 Pretvorba barvne v sivinsko sliko ... 9

4.2 Detekcija besedila ... 10

4.2.1 Iskanje robov ... 10

4.2.1.1 Konvolucija ... 11

4.2.1.2 Iskanje robov na podlagi odvoda ... 13

4.2.1.3 Sobelov operator ... 16

4.3 Lokalizacija besedila ... 17

4.3.1 Postopki lokalizacije ... 17

4.3.1.1 Metoda z upoštevanjem območij (Region - based method) ... 17

4.3.1.2 Metode, ki upoštevajo teksturo (Texture-based methods) ... 18

4.3.2 Filtriranje in inicializacija postopka lokalizacije besedila ... 18

4.3.2.1 Izboljšava koordinat lokaliziranega besedila ... 19

4.3.3 Sledenje in zaznavanje besedila v MPEG video signalu ... 20

4.3.4 Sledenje besedil v video signalu (Text Tracking in Videos) ... 21

4.3.4.1 Izločanje vektorja gibanja (Motion Vector Extraction) ... 22

4.3.4.2 Algoritem sledenja (The Tracking Algorithm) ... 24

4.4 Segmentacija in binarizacija ... 25

4.4.1 Segmentacija in binarizacija besedila v sliki ... 26

4.4.2 Barvni model RGB ... 26

24-bitne barve ... 27

4.4.3 Postopek segmentacije za neurejeno prilagodljivo besedilo (Adaptive fuzzy text segmentation method) ... 27

4.4.3.1 Izboljšanje ločljivosti ... 28
4.4.3.2 Izločanje in normalizacija značilk ... 29
4.4.3.3 Določitev začetnega števila grozdov ... 30
4.4.3.4 Inicializacija centroidov grozda ... 31
4.4.3.5 Neurejeno prilagodljivo grozdenje ... 32
4.4.3.6 Binarizacija in identifikacija besedila ... 33
4.4.3.7 Izboljšanje slike besedila ... 36
4.5 Izločanje značilk (feature extractor) ... 36
4.6 Tehnologija optičnega prepoznavanja znakov (OCR) ... 37
4.6.1 Zgodovinski razvoj ... 37
4.6.2 Delitev sistemov za prepoznavanja znakov ... 38
4.6.2.1 Delitev glede na namen ... 38
4.6.2.2 Delitev glede na vrsto zapisa ... 38
4.6.3 Postopek prepoznavanja znaka ... 39
4.6.3.1 Pred obdelava slike ... 40
4.6.3.2 Končna obdelava ... 40
5 DSP RAZVOJNI SISTEM BLACKFIN 42
6 IMPLEMENTACIJA SISTEMA ZA IZLOČANJA BESEDILA V VIDEO SIGNALU 45
6.1 Postopek procesiranja video signala ... 45
6.2 Postavitev strojne opreme za procesiranja video signala ... 46
6.3 Opis strojne opreme ... 49
6.3.1 Zunanji TV tuner AverMedia AVerTV Hybrid Ultra USB ... 49
6.3.2 Uporaba SDRAMa za delo z video podatki ... 50
6.4 Programski del ... 51
6.4.1 Razvojno orodje Visual DSP 5.0 ... 51
6.4.2 Način nalaganja programa za procesiranje video signala ... 51
6.4.3 Analiza programskega dela modula za zaznavanje robov v video signalu ... 53
6.4.4 Analiza modula za zaznavanje robov v video signalu ... 53
6.4.5 Prikaz poteka klicanja funkcij modula za zaznavanje robov ... 54
6.4.5.1 Opis delovanja poteka programske kode za zaznavanje robov ... 56
7 REZULTAT 61
8 SKLEP 62
9 VIRI, LITERATURA 63
10 PRILOGE 64
10.1 Naslov študenta ... 64
10.2 Življenjepis ... 65

KAZALO SLIK

slika 1: Blokovna shema sistema za razpoznavanje besedila v digitalnem video signalu 3
Slika 2: Različni primeri kompleksnejših slik z umetnim besedilom [1]. 5
Slika 3: Različni primeri zapletenih slik s prizori besedila v sceni [1]. 5
Slika 4: Proces izločanja besedila iz slike ali video signala na funkcionalnem nivoju [1]. 7
Slika 5: Arhitektura sistema izločanja besedila iz slike ali video signala [1] 8
Slika 6: Sivinska slika. 9
Slika 7: Psevdo koda za pretvorbo barvne slike v sivinsko 1 [6]. 9
Slika 8: Psevdo koda za pretvorbo barvne slike v sivinsko 2 [6]. 10
Slika 9: Spremembe v intenzitetah med slikovnimi točkami [6]. 11
Slika 10: Primer apliciranja konvolucijske maske [6] 12
Slika 11: Od leve proti desni: maska za glajenje, maska za ostrenje, maska za prikaz 13
Slika 12: Odvod intenzitet točk na robu 14
Slika 13 : Maski Sobelovega operatorja. Na levi maska Gx, na desni maska Gy. 16
Slika 14: Slika robov, dobljena s Sobelovim operatorjem [6] 16
Slika 15: Vertikalni in horizontalni operator razširitve (»dilation«). 19
Slika 16: Uporaba algoritma SDB [1]. 20
Slika 17: Pristop sledenja besedila v toku video okvirjev. 21
Slika 18: Algoritem zaznavanja/lokalizacije besedila v povezavi z algoritmom sledenja. 22
Slika 19: Vizualizacija MPEG vektorja gibanja [1]. 23
Slika 20: Psevdo koda »intra-GOP« metode sledenja. 25
Slika 21: Mešanja barv. 27
Slika 22: Diagram postopka segmentacije za neurejeno prilagodljivo besedilo 28
Slika 23: Eno dimenzionalna funkcija kubične interpolacije. 29
Slika 24: Vpliv AFC na končni rezultat segmentacije besedila. 33
Slika 25: Predstavljeni pravokotnik omejuje izločene povezane komponente za vsak kandidat besedila v sliki (C = 3) 34
Slika 26: Psevdo koda algoritma za klasificiranje slik besedila. 35

Slika 27: Vpliv postopkov izboljšanja na končni rezultat procesa segmentacije slike besedila. 36
Slika 28: Osnovni koraki OCR razpoznavanja..... 40
Slika 29: Blokovna shema razvojnega okolja Blackfin..... 42
Slika 30: Analog Devices BF561-EZKit..... 43
slika 31: Video dekodirnik ADV7183B..... 44
slika 32: Video kodirnik ADV7179. 44
Slika 33: Postopek procesiranja video signala. 45
Slika 34: Priključitev napajanja in USB vmesnika na BLACKfin DSP platformo..... 46
Slika 35: Philips DVDR 75. 47
Slika 36: Scart-RCA/S-video/DIN. 47
Slika 37: Povezava kabla scart-RCA/S-video/DIN z DVD predvajalnikom. 47
Slika 38: Povezava na razvojno ploščo BLACKfin BF-561 DSP..... 48
Slika 39: TV tuner na S – video način 48
Slika 40: TV tuner na »composite« način. 49
Slika 41: Zunanji TV tuner AverMedia AVerTV Hybrid Ultra USB. 50
Slika 42: Blokovna shema. 50
slika 43 : Nalaganje kode v SDRAM. 51
Slika 44: Program teče. 52
Slika 45: Nastavitev AVerTV 6 za prikaz video signala na monitorju. 52
Slika 46: Glavna funkcija »main« programa za zaznavanje robov. 53
Slika 47: VideoInEdgeDetection_sistem..... 54
Slika 48: VideoInEdgeDetection_Callbacks. 55
Slika 49: VideoInEdgeDetection_Buffers. 55
Slika 50: Procesirani video signal 61
Slika 51: Prvotni video signal 61

UPORABLJENE KRATICE

DVD	digitalni pomnilniški medij (Digital Versatile Disc)
USB	univerzalno serijsko vodilo (Universal Serial Bus)
DSP	digitalno procesiranje signalov (Digital Signal Processing)
JPEG	rastrski slikovni format (Joint Photographic Experts Group)
MPEG	video format (Moving Picture Experts Group)
RGB	barvni model (red, green, blue)
TL	lokalizacija besedila (Text localization)
TD	zaznavanje besedila (Text Detection)
SDB	algoritem, ki temelji na empirični ugotovitvi (Standard Deviation Based)
TT	sledenje besedila (Text Tracking)
TSB	segmentacija in binarizacija besedila (Text segmentation and Binarization)
AFC	adaptivno mehko klasificiranje (Adaptive fuzzy clustering)
CR	razpoznavanje znakov (Character Recognition)
OCR	optična razpoznava znakov (Optical Character Recognition)
ICR	inteligentno razpoznavanje znakov (Inteliligent character recognition)
OMR	razpoznavanje označevanj (Optical mark recognition)

1 UVOD

Dandanašnja računalniška tehnologija omogoča povsem digitalno obdelavo analognih signalov. Bistvo digitalnega procesiranja signalov (DSP) je, da analogni signal takoj prevedemo v digitalno obliko in ga nato procesiramo na računalniku. Vse potrebne filtre, modulatorje in demodulatorje, fazno sklenjene zanke in druga vezja nadomestimo z ustreznimi matematičnimi enačbami, ki jih nato rešuje računalnik v realnem času: sproti za vsak otipek signala z A/D pretvornika.

1.1 Ideja diplomskega dela

Ideja diplomskega dela je podrobna analiza procesa razpoznavanja besedila v digitalnem video signalu, v realnem času ter zasnova takšnega sistema na strojni opremi z izbranim DSP procesorjem. Za procesiranje video signala smo uporabili razvojno ploščo Blackfin ADSP – BF561, ki spada v družino 16 oz 32 bitnih mikroprocesorjev. Za to družino je značilno, da imajo vgrajen digitalni signalni procesor (DSP). Takšne platforme potrebujejo majhno moč, imajo poenoteno procesorsko enoto, ki lahko zažene operacijski sistem, medtem ko se ukvarja s kompleksnimi numeričnimi metodami, kot je na primer video kodiranje v realnem času.

1.2 Struktura diplomskega dela

Za lažje razumevanje diplomskega dela je diploma razdeljena na deset poglavij.

Prvo poglavje pričenjamo z uvodom, kjer na splošno opišemo digitalno procesiranje signalov in samo idejo sistema.

V drugem poglavju nato predstavimo zasnova sistema ter opišemo kako in kaj vse bomo uporabili pri izvedbi diplomskega dela.

Tretje poglavje zajema predstavitev problema obstoja besedila v sliki ali videu,

Četrto poglavje podaja opis celotnega procesa izločanja besedila iz video signala.

Peto poglavje predstavlja razvojni sistem Blackfin DSP.

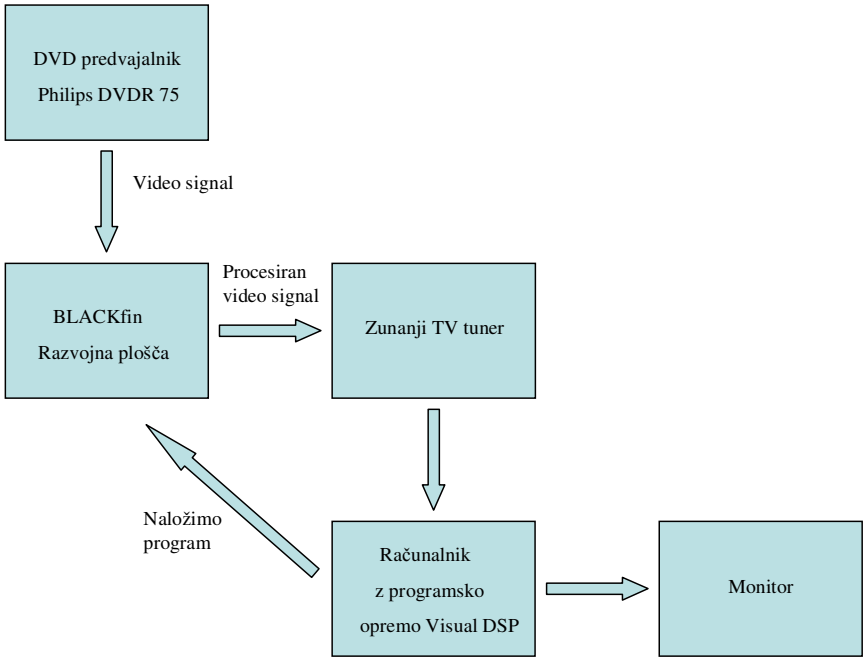
Šesto poglavje se ukvarja z implementacijo zaznavanja besedila v digitalnem video signalu, kjer opišemo vzpostavitev razvojnega okolja in opišemo programski del.

Sedmo poglavje prikazuje rezultat, osmo poglavje je namenjeno sklepom diplomskega dela.

V devetem poglavju je našteta literatura, v zadnjem, desetem poglavju pa so podane priloge k diplomski nalogi, ki zaključijo celotno delo.

2 ZASNOVA SITEMA

V diplomskem delu smo zasnovali sistem prepoznavanja besedila v digitalnem video signalu, ki teče na razvojni plošči Blackfin. Video signal pripeljemo iz DVD predvajalnika v video dekodirnik ADV7183, ki se nahaja na razvojni plošči BLACKfin ADSP-BF561 EZ-Kit. Ko se procesiranje izvede, se izhodni signal iz ADV7179 video kodirnika, ki se prav tako nahaja na razvojni plošči BLACKfin, prenese preko zunanjega TV tunerja na računalnik. Programski del za razpoznavanje besedila v digitalnem video signalu bo spisan v programskem okolju Visual DSP, s programskim jezikom C. V blokovni shemi sistema, ki je prikazan na sliki 1, so podane osnovne sestavne komponente sistema.



slika 1: Blokovna shema sistema za razpoznavanje besedila v digitalnem video signalu

Komponente sistema so:

- DVD predvajalnik Philips DVDR 75
- BLACKfin ADSP-BF561 EZ-Kit

- AverMedia AVerTV Hybrid Ultra USB
- Osebni računalnik z nameščeno programsko opremo Visual DSP

3 BESEDILO V SLIKI IN VIDEO SIGNALU

Besedilo v sliki ali video signalu razvrščamo v dve glavni skupini in sicer glede na njegov izvor:

- umetno besedilo (v nadaljevanju tudi kot zajemanje besedila ali prekrivanje besedila),
- tekst v sceni (v nadaljevanju tudi kot grafično besedilo)

Umetno besedilo (slika 2) je na sliko dodano naknadno (npr. ime nekoga med pogovorom, ime govorca itd.). V nasprotju z umetnim besedilom je tekst v sceni (slika 3) del scene (npr. imenovanja produkta med reklamnim premorom itd.) in ga je ponavadi težje izločiti zaradi različnih poravnav, velikosti, vrste pisave, zapletenosti ozadja in distorzije. Poleg tega nanj pogosto vplivajo odstopanja v sceni in parametri kamere, kot so osvetlitev, izostritev, gibanje, itd.

Slika 2: Različni primeri kompleksnejših slik z umetnim besedilom [1].

Slika 3: Različni primeri zapletenih slik s prizori besedila v sceni [1].

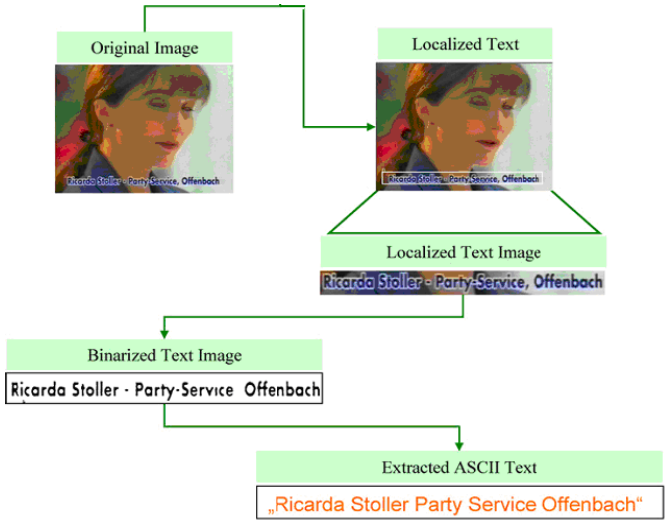
Pri zaznavanju besedila v sliki in video signalu moramo upoštevati naslednje značilnosti:

- Geometrijske lastnosti (velikosti pisave, razporeditev, razdalja znakov),
- Razmerje barv (sivina, enobarvna, večbarvna),
- Gibanje (statično, linearno, prosto gibanje),
- Robovi/kontrast (intenzivni robovi (kontrast) na mejah besedila)
- Stiskanje (JPEG, MPEG – stisnjena slika)
- Ozadje.

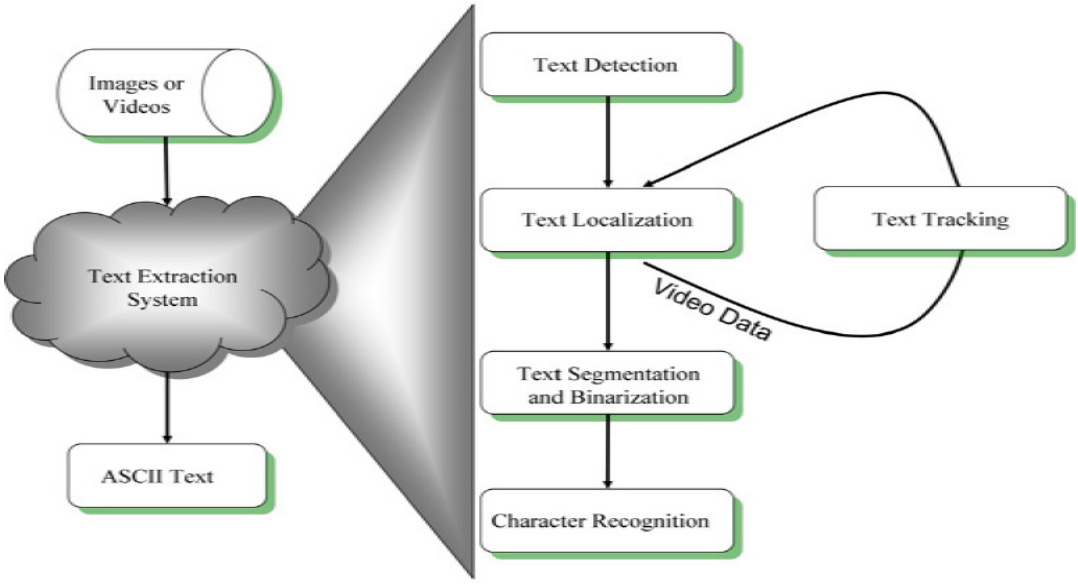
4 PROCES IZLOČANJA BESEDILA

Izločanje besedila (text extraction) je kompleksen problem, ki vključuje naslednje korake procesiranja:

- zaznavanje besedila (text detection),
- lokalizacija besedila (text localization),
- sledenje besedila (text tracking),
- segmentacija in binarizacija besedila (text segmentation and binarization),
- optično razpoznavanje znakov (optical character recognition - OCR).



Slika 4: Proces izločanja besedila iz slike ali video signala na funkcionalnem nivoju [1].



Slika 5: Arhitektura sistema izločanja besedila iz slike ali video signala [1].

4.1 Opis iskanja štirirobnih objektov

Predno se lotimo iskanja besedila v sliki, je potrebno pred tem sliko obdelati na naslednji način:

- Na začetku sliko po potrebi zmanjšamo, da tako zmanjšamo čas iskanja objektov.
- Barvno sliko nato pretvorimo v sivinsko.
- Na sivinski sliki poiščemo robove (poglavje 4.2.1).

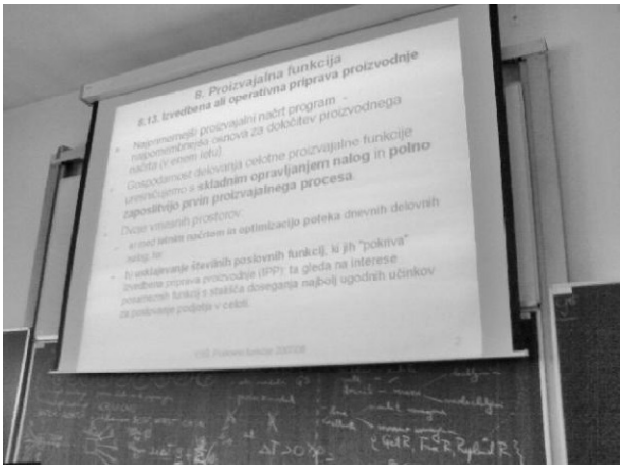
4.1.1 Spreminjanje velikosti slike

Da določimo štirirobne objekte na sliki je pomembno, da sliko zmanjšamo na določeno velikost, kajti program za iskanje štirirobnih elementov bo deloval bistveno hitreje. Pomembno je, da ohranimo razmerje stranic slike, zato moramo enakomerno spreminjati velikost slike. Enakomernemu spreminjanju velikosti slike pravimo tudi skaliranje. Zaradi zmanjševanja slike izgubimo nekaj natančnosti prikaza najdenih štirirobnih objektov, kjer lahko to opazimo pri koordinatah odkritih površin. Ta natančnost je v praksi sicer zelo majhna, saj sliko zmanjšamo le za majhen faktor. Sliko zmanjšamo z zmanjšanjem števila slikovnih elementov, iz katerih je slika sestavljena. Zaradi tega se izgubijo nekatere informacije v sliki, kar se pa izkaže tudi kot pozitivno pri iskanju večjih površin slike.

Zaradi zmanjšanja slike se zmanjša tudi kompleksnost slike in zaradi tega je celotna iskana oblika štiririobnega objekta še bolj izrazita [6].

4.1.2 Pretvorba barvne v sivinsko sliko

Pri iskanju objektov na sliki nas ne zanimajo barve slikovnih točk, temveč samo njihove intenzitete. **Algoritem Sobel** [4], ki ga uporabljamo za iskanje robov, temelji na iskanju gradientov intenzitet. Ker pri iskanju robov ne potrebujemo barvne informacije o točkah, je barvno sliko smiselno najprej pretvoriti v sivinsko sliko (slika 6), da s tem poenostavimo nadaljnji postopek analize slike.



Slika 6: Sivinska slika.

Sivinsko sliko dobimo s pretvorbo barvne slike s povprečjem vrednosti barvnih točk [6]. Vrednosti novih sivinskih slikovnih točk dobimo tako, da seštejemo vrednosti barvnih kanalov istih točk in dobljeno vsoto delimo z njihovim številom.

```
/** psevdo koda za pretvorbo barvne slike v sivinsko 1 **/

za vsako slikovno točko P na sliki
   P.intenziteta = (P.rdeča + P.zelena + P.modra)/3
```

Slika 7: Psevdo koda za pretvorbo barvne slike v sivinsko 1 [6].

Nekoliko boljše rezultate dobimo z upoštevanjem intenzitetne uteži posameznih kanalov. Vsak barvni kanal ne prispeva enakega deleža končni vrednosti intenzitete slikovne točke. Modri kanal ima med vsemi kanali največji kontrast, Zeleni pa najmanjšega. Standardizirane uteži za model RGB so 0,299 za rdeči, 0,587 za modri in 0,114 za zeleni kanal.

```
/** psevdo koda za pretvorbo barvne slike v sivinsko 2 **/

za vsako slikovno točko P na sliki
   P.intenziteta = P.rdeča  * 0,299 +
                   P.zelena * 0,587 +
                   P.modra  * 0,114
```

Slika 8: Psevdo koda za pretvorbo barvne slike v sivinsko 2 [6].

Na tej stopnji ne govorimo več o barvah točk, temveč o njihovih intenzitetah.

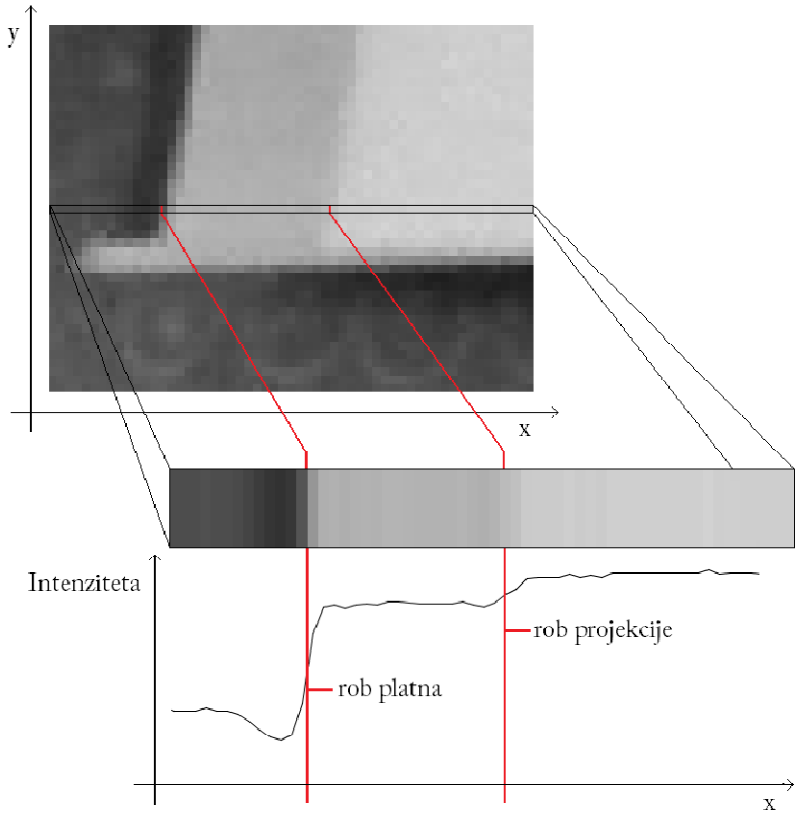
4.2 Detekcija besedila

Na začetku, ko še nimamo informacije ali vhodna slika sploh vsebuje besedilo, moramo uporabiti postopek detekcije besedila (text detection), ki poda odgovor ali besedilo dejansko obstaja na sliki/video ali ne. Različni pristopi predpostavljajo, da nekateri video okvirji ali slike vsebujejo besedilo. To je osnovna predpostavka na primer za skenirane slike. Ponavadi je število video okvirjev, ki vsebujejo besedilo precej manjše, kot pa je število video okvirjev, ki ne vsebujejo besedila. Postopek detekcije besedila, poizkuša odkriti prisotno besedilo v dani sliki ali video okvirju. Poznamo več algoritmov za zaznavanje besedila, kot so: Sobel, Roberts, Prewitt ali Canny operator, itd. [1].

4.2.1 Iskanje robov

Pri prepoznavanju besedila v digitalnem video signalu je ena najpomembnejših operacij iskanje robov. Robovi predstavljajo meje zaključenih oblik, ki so lahko npr. tudi prezentacijske projekcije, kot je to prikazano na sliki 9. Lahko pa predstavljajo tudi ločnico med svetlobo in senco, ki pada na posamezno površino. S pretvorbo slike smo dobili sliko, kjer so vidni samo robovi. Ostali podatki so izločeni, ker so za nas nepomembni v nadaljni

analizi slike. Najdene robove lahko prikažemo z dvodimenzionalno tabelo, kjer imajo polja z robovi vrednost ena, ostala polja pa vrednost nič. Zaradi boljše vizualne predstave in poenostavitve nadaljnih postopkov pri analizi slike, jih velikokrat predstavimo kar z enobitno sliko. Piksli, ki predstavljajo robove na sliki so bele barve, vse ostalo pa črne barve.



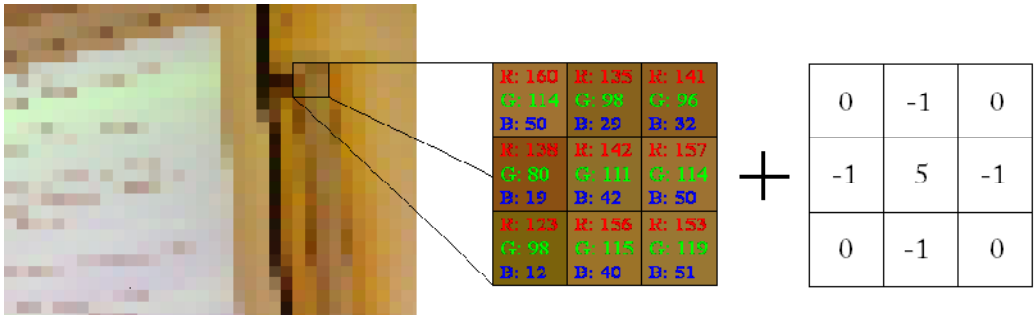
Slika 9: Spremembe v intenzitetah med slikovnimi točkami [6].

4.2.1.1 Konvolucija

Konvolucija je postopek, pri katerem s pomočjo konvolucijskih mask različnih velikosti spremenimo lastnosti vhodne slike. Konvolucijsko masko položimo na sliko in potujemo skozi vsako slikovno točko (piksel) na njej. Nato vzamemo slikovne točke, ki ležijo pod masko in izračunamo novo vrednost za sliko. Njihove vrednosti uravnotežimo glede na polja v maski, nato pa jih med seboj seštejemo, da dobimo novo vrednost slikovne točke (slika 8). Pri barvnih slikah je potrebno postopek ponoviti za vsak barvni kanal. Če je vsota vseh polj v konvolucijski maski enaka ena, se ohranja povprečna intenziteta slikovnih točk. Če je skupna vsota manjša dobimo temnejšo, če je večja, pa svetlejšo sliko. Kadar skupna

vsota polj v konvolucijski maski ni enaka ena, mi pa bi radi ohranili intenzitete slikovnih točk, dobljene vrednosti pomnožimo s faktorjem, ki nevtralizira njihove vrednosti. Primer maske, pri kateri moramo ohraniti intenzitete točk, je **maska za glajenje** (slika 11). Pri konvoluciji moramo paziti, da novih vrednosti ne zapisujemo nazaj v vhodno sliko. V tem primeru bi zaradi spremenjenih vhodnih slikovnih elementov dobili napačne vrednosti.

Časovna kompleksnost postopka je ovrednotena z O(p * $n2$) .



Slika 10: Primer apliciranja konvolucijske maske [6].

Z računanjem novih vrednosti za vsak barvni kanal dobimo novo vrednost slikovne točke, kot sledi [6]:

$$R = (0 \times 160) + (-1 \times 135) + (0 \times 141) + (-1 \times 138) + (5 \times 142) + (-1 \times 157) + (0 \times 123) + (-1 \times 156) + (0 \times 153) = 124 \quad (4.1)$$

$$G = (0 \times 114) + (-1 \times 98) + (0 \times 96) + (-1 \times 80) + (5 \times 111) + (-1 \times 114) + (0 \times 98) + (-1 \times 115) + (0 \times 119) = 148 \quad (4.2)$$

$$B = (0 \times 50) + (-1 \times 29) + (0 \times 32) + (-1 \times 19) + (5 \times 42) + (-1 \times 50) + (0 \times 12) + (-1 \times 40) + (0 \times 51) = 72 \quad (4.3)$$

S konvolucijskimi maskami lahko sliko izostrimo tako, da povečamo kontrast med sosednjimi ležečimi točkami na sliki (angl. sharpening). S tem povečamo število vidnih podrobnosti na njej. Lahko naredimo tudi obraten postopek in sliko zameglimo (angl. blur). V tem primeru sliko »zamažemo« tako, da zmešamo barve med sosednje ležečimi

točkami. Na sliki lahko tudi ustvarimo tridimenzionalni senčni videz slike in jo prikažemo kot relief (angl. embossing). Med pomembnejše naloge konvolucije sodi iskanje robov, ki je podrobneje opisano v naslednjem poglavju.

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

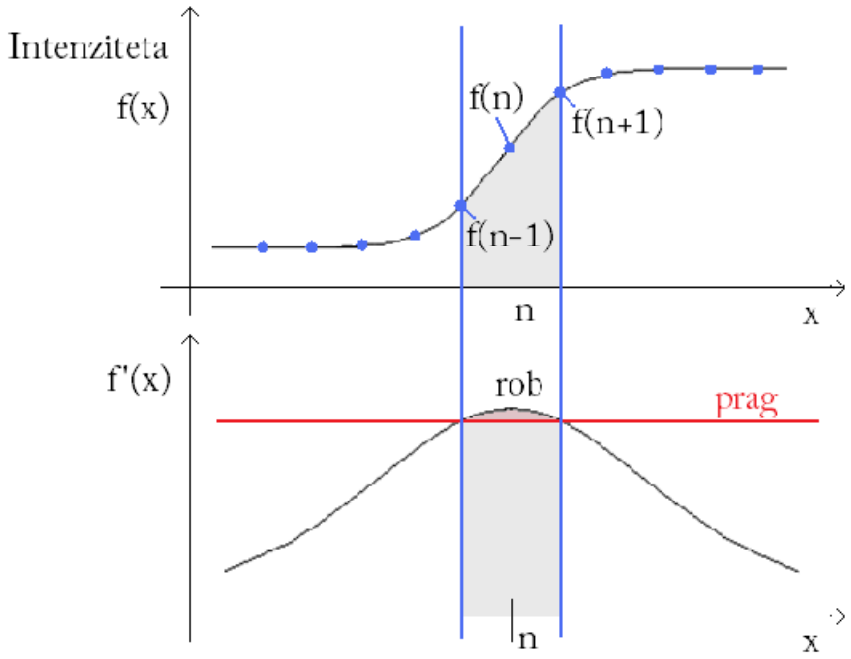
0	-1	0
-1	5	-1
0	-1	0

-1	0	0
0	0	0
0	0	1

Slika 11: Od leve proti desni: maska za glajenje, maska za ostrenje, maska za prikaz reliefa [6].

4.2.1.2 Iskanje robov na podlagi odvoda

Formalno so robovi na sliki predstavljeni kot nenadne spremembe v intenzitetah sosednjih slikovnih točk. Čim hitrejši je prehod v intenzitetah teh točk (gradient), večja je verjetnost, da se nahajajo na robu. Hitrost spremembe v intenzitetah dobimo tako, da izračunamo odvod. Če je vrednost odvoda večja od nekega vnaprej določenega praga, potem smo na tem mestu naleteli na rob (slika 12) [6].



Slika 12: Odvod intenzitet točk na robu

Odvod f'(x) je definiran kot sprememba vrednosti funkcije f(x) pri spremembi njenega argumenta Δx:

$$f'(x) = \frac{d(f(x))}{dx} \tag{4.4}$$

Približno vrednost odvoda f'(x) pri vrednosti argumenta x = n izračunamo po naslednji enačbi:

$$f'(n) = \frac{d(f(n))}{dn} = \frac{f(n+1)-f(n-1)}{(n+1)-(n+1)} = 0.5 * (f(n+1) - f(n-1)) \tag{4.5}$$

Slikovna funkcija f(x,y) nam vrne intenziteto slikovne točke na x in y koordinatah bitne slike. Ker je slika dvodimenzionalno polje točk, moramo z uporabo parcialnih odvodov izračunati približka gradientov robov v smereh x (Gx) in y (Gy). S prvim zaznamo navpične robove, z drugim pa vodoravne robove.

Odvod po x-u:

$$\frac{\partial f(x,y))}{\partial x} = Gx = 0.5 * (f(x+1,y) - f(x-1,y)) \tag{4.6}$$

Odvod po y-u:

$$\frac{\partial f(x,y))}{\partial y} = Gy = 0.5 * (f(x, y+1) - f(x, y-1)) \tag{4.7}$$

Približke teh odvodov lahko na bitni sliki dobimo z apliciranjem dveh konvolucijskih mask (z levo dobimo Gx, z desno pa Gy):

$$0.5 * \begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \quad 0.5 * \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{4.8}$$

Dobljeni vrednosti obeh gradientov (odvodov) na koncu združimo, da dobimo absolutno intenziteto gradienta. Absolutno vrednost gradienta na slikovni funkciji $\nabla f\ x, y$ dobimo na naslednji način:

$$g(x,y) = \sqrt{Gx^2 + Gy^2} = \nabla f(\mathrm{x,y}) = \sqrt{\left(\frac{\partial f(x,y)}{\partial x}\right)^2 + \left(\frac{\partial f(x,y)}{\partial y}\right)^2} \tag{4.9}$$

Zaradi hitrejšega računanja ponavadi uporabimo kar približek:

$$g(x,y) = |Gx| + |Gy| \tag{4.10}$$

Če je absolutna intenziteta gradienta večja od praga, potem se slikovna točka, na koordinatah (x,y), nahaja na robu:

$$e(x,y) = \begin{cases} 1, & g(x,y) > prag \\ 0, & sicer \end{cases} \tag{4.11}$$

Vrednost praga moramo vnaprej določiti. Če določimo visok prag, bomo dobili samo najbolj izrazite robove, se pravi tiste z največjim naklonom (gradientom) in s tem največjo vrednostjo odvoda. Ob izbiri nizkega praga pa bomo velikokrat poleg pravih dobili tudi robove, ki ne predstavljajo meje dveh neenakih regij v sliki, temveč so lahko posledice motenj (šuma) na sliki. Šum je na sliki viden kot niz drobnih pikic, ki so bolj vidne v temnejših delih slike. Če je bolj izrazit, je lahko navzoč tudi na celotni sliki. Primera

detektorjev robov, ki temeljita na računanju odvoda, sta **Robertsov** in **Sobelov operator**, ki smo ga uporabili tudi v diplomskem delu.

4.2.1.3 Sobelov operator

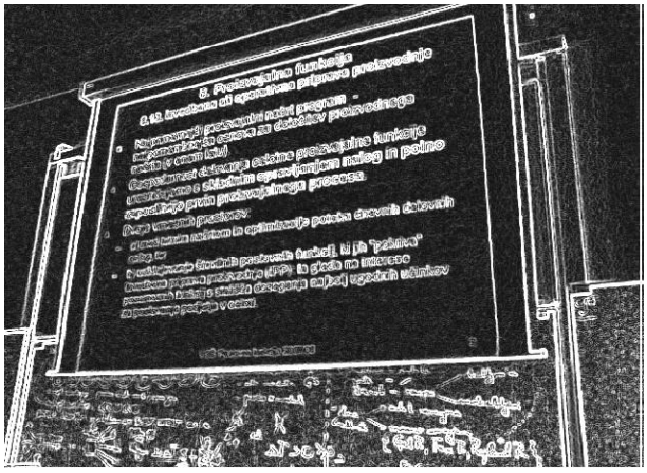
Sobelov operator išče robove z ugotavljanjem gradientov. Uporablja dve 3x3 konvolucijski maski.

1	0	-1
2	0	-2
1	0	-1

1	2	1
0	0	0
-1	-2	-1

Slika 13 : Maski Sobelovega operatorja. Na levi maska Gx, na desni maska Gy.

Operator zaradi večjih mask bolj gladi sliko, kar ga naredi manj občutljivega na šum. Večji maski povzročita, da operator lahko zazna tudi manjše spremembe v intenzitetah sosednjih slikovnih točk in manj izrazite robove. Intenzitete najdenih robov so nekoliko večje, zaradi večje maske pa so ti tudi nekoliko širši.



Slika 14: Slika robov, dobljena s Sobelovim operatorjem [6].

4.3 Lokalizacija besedila

Cilj postopka lokalizacije besedila je, da se loči ozadje od besedila v sliki ali video okvirja. Za postopek lokalizacije je pomembno tudi, da je čim hitrejši in v najboljšem primeru ne določi območje besedila kot ozadje. Nadaljni cilj postopka je, da izločimo posamezne vrstice besedila tako, da se lahko velikosti vrstice besedila normalizirajo. To lahko izvedemo preko detekcije robov in z ekstrakcijo tistih blokov, ki so kandidati za besedilo. Pri postopku lokalizacije besedila (text localization - TL) pripeljemo na vhod sliko ali video okvir, ki smo ga predhodno obdelali s postopkom detekcije besedila in združimo tista območja besedil, ki pripadajo enakim kandidatom besedila. Končna naloga postopka je, da določi natančne koordinate položaja besedila [1].

4.3.1 Postopki lokalizacije

V nadaljevanju predstavljamo dve metodi, s katerima lahko zaznamo in lokaliziramo besedilo v sliki ali v video posnetku.

4.3.1.1 Metoda z upoštevanjem območij (Region - based method)

Metode z upoštevanjem območij delimo v dva razreda: takšne, ki upoštevajo povezovanje komponent (CC) (connected component CC-based) in na takšne, ki upoštevajo robove. Te metode se izvajajo od spodaj navzgor (bottom-up) – najprej se namreč identificira osnovne (majhne) pod-strukture, kot so na primer CCji ali robovi, nato pa se izvaja združevanje le-teh v večje strukture, vse dokler niso zaznana vsa področja besedil. V primeru metod, ki upoštevajo povezovanje komponent (CC), se osnovni elementi ustvarijo z uporabo podobnosti sosednjih slikovnih pik v barvnih ali črnobelih odtenkih, medtem ko metoda na osnovi robov izkorišča visok kontrast med besedilom in ozadjem. Nekateri pristopi uporabljajo tudi kombinacijo obeh metod.

- **Metoda CC**

  Temelji na uporabi pristopa od spodaj navzgor, pri zaporedoma združujemo manjše strukture, kot so CC ali robovi, v večje strukture, dokler ne zaznamo vsa območja besedil. Pri združevanju komponent besedil uporabljamo geometrično analizo, ki sloni na prostorski ureditvi komponent. Le-ta pomaga izločiti tiste komponente, ki ne predstavljajo besedila in določiti meje območij z besedilom.
- **Metode, ki upoštevajo robove (Edge-based methods)**

  Ta metoda se osredotoča na visok kontrast med besedilom in ozadjem, zazna nastale robove iz obrisov besedila in jih potem združi, če je to mogoče.

4.3.1.2 Metode, ki upoštevajo teksturo (Texture-based methods)

Metode, ki upoštevajo teksturo, temeljijo na opažanju, da besedilo na sliki izkazuje določene teksturne lastnosti, ki jih lahko uporabimo pri ločevanju besedila od ozadja slike. Gabor filtre, valčke (Wavelets), hitro Fourierjevo transformacijo in podobne običajno uporabimo za izločanje teksturnih lastnosti območij z besedilom na sliki.

V poglavju 4.3.2 bomo podrobneje predstavili eno od metod za lokalizacijo besedila na osnovi strukture.

4.3.2 Filtriranje in inicializacija postopka lokalizacije besedila

Tako imenovano označeno sliko (marked image) ustvarimo na podlagi klasificiranih blokov slikovnih pik v grozdu besedila. Morebitne vrzeli zapolnimo z uporabo matematičnega morfološkega operatorja, imenovanega »dilation« (razširitev). Za razširitev označene slike lahko v navpični in vodoravni smeri uporabimo dva različna strukturna elementa (slika 15). Razširjene označene slike nato oblikujemo kot neusmerjen graf, pri čemer vozlišča predstavljajo bloke besedila, robovi pa predstavljajo dejstvo, da so ti bloki besedila sosedi. Nato povezane komponente izločimo z uporabo iskalnega algoritma «depth-first-search«. Posamezne ločene komponente zavržemo kot ozadje. Na koncu upoštevamo najbolj leve in najbolj desne ter zgornje in spodnje koordinate, ki določajo

začetni pravokotnik. Te koordinate tako predstavljajo začetno lokalizacijo besedila, ki pa jo lahko še dodatno izboljšamo z nadaljnjimi koraki procesiranja.

(a) 5x1 element strukture za navpično razširitev. (b) 3x6 element strukture. za vodoravno razširitev

Slika 15: Vertikalni in horizontalni operator razširitve (»dilation«).

4.3.2.1 Izboljšava koordinat lokaliziranega besedila

Kot rezultat začetne lokalizacije lahko dobimo več vrstic besedila skupaj. Za iskanje koordinat posamezne vrstice besedila, bomo zato v nadaljevanju uporabili enega izmed naslednjih algoritmov:

Algoritem SDB (Standard deviation based) [1] temelji na empirični ugotovitvi, da je razdalja med ostrimi robovi v navpični smeri bolj urejena v primeru vrstice z besedilom, kot v primeru ozadja z določeno teksturo. Posledično pričakujemo, da bo za vrstico z besedilom značilen nižji standardni odklon razdalje robov. Nad že dobljenim poljem z besedilom »textBox«, izvedemo dani algoritem na naslednji način:

1. Za vsako vrstico i, $line_i \in textBox$: izračunamo razdalje ($D_i[\ ]$) med zaporednimi intenzivno močnimi slikovnimi elementi robov, ki so prisotni v pasu slike LH.

2. Ovrednotimo standardno deviacijo $stDev_i$ razdalj $D_i[]$, ločeno za vsako vrstico i.

3. Razdelimo izhodiščno polje besedila »textBox« na dva razreda z manjšimi polji besedila textBox_k, tako da vsa polja textBox_k, ki spadajo v en ali drugi razred, izpolnijo enega od naslednjih pogojev:

   (a) $\text{textBox}_k \in class_1$, potem $\text{stDev}_i < \text{MaxStDev}, \forall\, \text{line}_i \in \text{textBox}_k$
   (b) $\text{textBox}_k \in class_2$, potem $\text{stDev}_i >= MaxStDev, \forall\, \text{line}_i \in \text{textBox}_k$

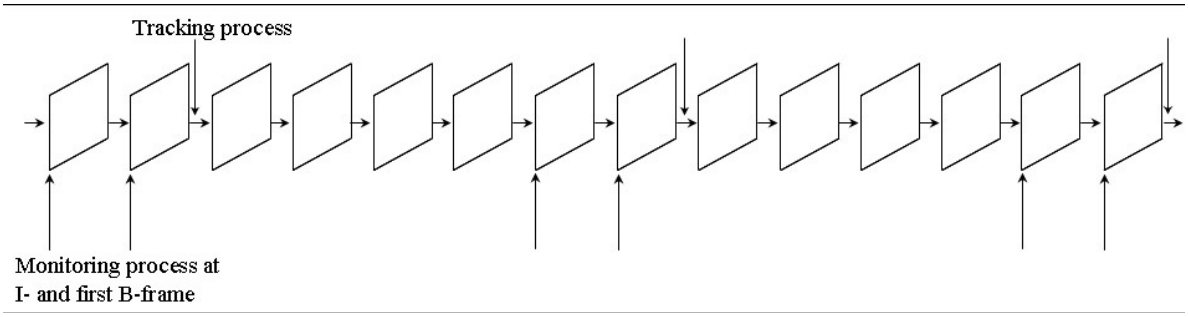
4. Zavržemo vse tiste bloke besedila, ki so uvrščeni v drugi razred, ker predstavljajo »lažen alarm«. Preostali bloki besedila iterativno večkrat preverjamo.

Na sliki 16 vidimo rezultat uporabe predstavljenega algoritma SDB.

Slika 16: Uporaba algoritma SDB [1].

4.3.3 Sledenje in zaznavanje besedila v MPEG video signalu

Na digitalnem video signalu se besedilo pogosto ohranja čez 30 ali več 100 okvirjev. V takem primeru potrebujemo dodatno še algoritme sledenja besedila (text tracking – TT), ki običajno izkoriščajo časovno komponento besedila v zaporedju video okvirjev. Ta postopek moramo vpeljati tudi zaradi zmanjšanja časa procesiranja že opisanih algoritmov [1].



Slika 17: Pristop sledenja besedila v toku video okvirjev.

Besedila, ki se pojavijo na video posnetku, so lahko statična ali gibljiva (linearno ali naključno) preko več okvirjev. Ker je lahko tudi ozadje statično ali dinamično, moramo upoštevati naslednje štiri možnosti:

- Statično besedilo in statično ozadje,
- Statično besedilo in dinamično ozadje,
- Dinamično ozadje in statično besedilo,
- Dinamično besedilo in statično ozadje.

Algoritem sledenja potrebujemo zato, da izkoristimo časovno omejen nastop besedila v video toku. Algoritem deluje neposredno nad MPEG signalom in uporablja za napovedovanje položaja besedila v naslednjem video okvirju informacijo MPEG vektorjev gibanja [1].

4.3.4 Sledenje besedil v video signalu (Text Tracking in Videos)

Kombinacija zaznavanja besedila in algoritma sledenja močno poveča učinkovitost delovanja sistema izločanja besedila v smislu časa procesiranja. Sledenje besedila v videoposnetkih lahko obravnavamo kot problem, ko se na okvirju pojavi več besedil in se teoretično lahko tudi gibajo v različnih smereh. Da bi zaznali besedila v različnih video okvirjih, v različnih časovnih točkah, moramo zaznavanje besedila in lokalizacijo izvesti v povezavi z algoritmom sledenja, kar prikazujemo tudi na sliki 18.

Algorithm 12: The text detection/localization and tracking approach for video data

Input: The video which will be analyzed
Output: The set of the found text boxes

```
1  k = 0;
2  Text detection and localization in frame_k;
3  while (k < (numberOfFrames − 1)) do
4      if ((frame_k! = P) and (frame_k! = B)) then
5          tBox_k[]=Text detection in frame_k;
6          if (k > 0) then
7              Text box tracking verification;
8          end
9      else
10         Text tracking: for each tBox_k[i] ∈ frame_k predict the new
           position tBox_{k+1}[i] in frame_{k+1} using Algorithm 13;
11     end
12     k = k + 1;
13 end
```

Slika 18: Algoritem zaznavanja/lokalizacije besedila v povezavi z algoritmom sledenja.

4.3.4.1 Izločanje vektorja gibanja (Motion Vector Extraction)

Če imamo zaporedje okvirjev tipa IBBP, potem vektorji gibanja v okvirju P kažejo na referenčne bloke v okvirju I. Originalni vektorji gibanja v okvirju P opisujejo gibanje, ki se pojavi znotraj teh okvirjev (IBB). Ker nas zanima gibanje med dvema zaporednima okvirjema, vektorji gibanja normalizirajo okvir na razdalji 1. Tako z normaliziranjem vektorja gibanja ocenjujemo gibanje med trenutnim in prejšnjim video okvirjem.

Slika 19: Vizualizacija MPEG vektorja gibanja [1].

Normalizacija je definirana kot:

$$frameDist = ||currentFrame - referenceFrame|| \tag{4.12}$$

$$normalizedVectorX = \frac{originalVectorX}{frameDist} \tag{4.13}$$

$$normalizedVectorY = \frac{originalVectorY}{frameDist} \tag{4.14}$$

Pri tem predstavlja *currentFrame* številko trenutnega okvirja, *referencaFrame* številko okvirja, v katerem so točke vektorja gibanja, *originalVectorX* in *originalVectorY* pa sta X in Y vrednosti vektorja gibanja, ki sta izločeni iz trenutnega okvirja (v popolni ločljivosti slikovnih pik). Uporaba vektorjev gibanja za sledenje besedila je težavna zaradi dejavnikov šuma in problema, kako določiti, kateri vektor gibanja opisuje gibanje besedila in kateri gibanje ozadja. Na sliki 19 lahko vidimo vizualno predstavitev vektorja gibanja v določenem okvirju. V tem video okvirju je giblje besedilo navpično, na gibajočem se ozadju. Mreža, ki jo vidimo na sliki, ponazarja makro bloke. Makro blok s samo eno točko ima vektor gibanja dolžine nič, medtem ko prazen makro blok predstavlja tiste kodirane bloke, kjer je kodiranje neodvisno od referenčnega okvirja in gibanja. Poleg tega lahko imajo vektorji gibanja v polju besedila tudi različno dolžino in smeri. To so nekatere dodatne težave pri algoritmu sledenja, ki jih moramo reševati.

4.3.4.2 Algoritem sledenja (The Tracking Algorithm)

Algoritem sledenja obravnava vse štiri že omenjene mogoče kombinacije, ki vključujejo gibanje ali mirovanje besedila in gibanje ali mirovanje ozadja. Vektorji gibanja so izločeni neposredno iz kompresiranega video signala MPEG in jih uporabljamo pri napovedovanju prihodnjega položaja besedila z le malo dodatnega časa računanja.

- **Sledenje, ki temelji na vektorjih gibanja (Motion Vector Based Intra-GOP Tracking) [1].**

  Za dano polje besedila $tBox_k$ [i], v okvirju $frame_k$, izvedemo postopek izločanja besedila v algoritmu 12 (korak 7), na naslednji način: najprej izračunamo ekvivalentne koordinate polja besedila $tBox_k$. Najbolj ujemajoč kvadrat s koordinatami makro bloka mbBox[(x'_0, y'_0), (x'_1, y'_1)], polja besedila tBox[(x_0, y_0), (x_1, y_1)] nato izračunamo z enačbo:

$$x'_i = Int\left(\frac{x_i+8}{16}\right) \tag{4.15}$$

$$y'_i = Int\left(\frac{y_i+8}{16}\right) \tag{4.16}$$

  Normalizirane vektorje gibanja (formula zgoraj) nato izločimo za vsak makro blok, katerega presek s poljem besedila je večji, kot predhodno postavljena vrednost praga (npr. 30%). Ta kriterij izberemo zato, da skušamo izločiti tiste vektorje gibanja, ki zelo verjetno opisujejo gibanje ozadja in ne gibanje besedila. Nato zberemo tisto množico vektorjev gibanja, ki napoveduje položaj polja besedila v naslednjem okvirju. To temelji na predpostavki, da množica vektorjev gibanja opisuje najverjetnejše gibanje celotnega polja besedila. Izbira uporabljenega načina (predmet z največjo frekvenco), pomaga, da odpravljamo nekatere šumne vektorje gibanja brez dodatnega filtriranja. V primeru polja besedila, ki je kodirano znotraj blokov, lahko predhodno gibanje polja besedila uporabimo pri napovedovanju

novega položaja. Za napovedovanje novega položaja besedila uporabljamo obrnjeni način delovanja vektorjev gibanja.

Algoritem na sliki 20 prikazuje algoritem predstavljene metode sledenja [1].

Algorithm 13: The motion vector based intra-GOP tracking algorithm

Input: The text box, $tBox_k[i]$ in the $frame_k$, which will be tracked

Output: The coordinates of the text box: $tBox_k[i]$ through the successive frames

1 Find text macroblocks in $frame_k$ which intersect more than a predefined ratio with the given text box;

2 Extract forward motion vectors from these text macroblocks;

3 Find the mode MV (the element with the maximum frequency) in the motion vector set;

4 Calculate the new position of the given text box, $tBox_{k+1}$: $tBox_{k+1} = tBox_k - MV$, where MV is the mode motion vector;

Slika 20: Psevdo koda »intra-GOP« metode sledenja.

4.4 Segmentacija in binarizacija

S segmentacijo razgradimo sliko posameznega bloka na slike posameznih znakov, ki jih kasneje ločeno razpoznavamo. Segmentacija poteka v več korakih. V bistvu je prvi korak segmentacije že omenjena dekompozicija dokumenta in tudi ostali postopki opisani pri predobdelavi slike. Pri razpoznavanju blokov je postopek običajno razdeljen na segmentacijo stolpcev, vrstic, besed in končno znakov. Segmentacija je otežkočena predvsem v primeru “zlepljenih” (*angl. touching characters, kissing fonts*) ali prelomljenih znakov ter delno prekrivajočih se območij znakov (*angl. kerning*), kar se dogaja predvsem pri ročni pisavi. Dodatni nivoji in algoritmi segmentacije poskušajo te napake odpraviti, saj v nasprotnem primeru pride do hipersegmentacije (več znakov združenih v enega) ali podsegmentacije znakov (razdelitev znakov). Metode segmentacije so na primer izmenjevanje vertikalne in horizontalne projekcije. Na splošno se delijo strategije segmentacije v tri skupine: *top-down, bottom-up, hibridne*. Ob segmentaciji se ponavadi izvajajo še procesi normalizacije znaka na standardno velikost in včasih tudi stanjšanje na

debelino enega slikovnega elementa (*angl. thinning algorithm*). Poudariti velja, da so avtorji precej neenotni pri opisih in delitvi procesov predobdelave in segmentacije slike. Dosledno se ne zgledujemo po nobenem od njih in smo procese razdelili, kot se nam je zdelo najbolj primerno.

4.4.1 Segmentacija in binarizacija besedila v sliki

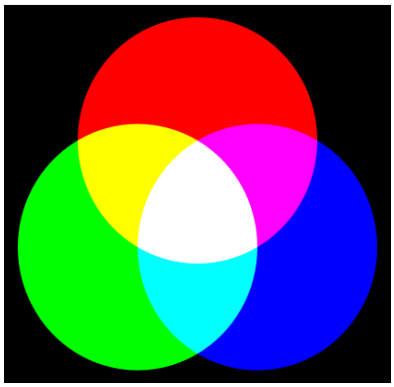
Segmentacija in binarizacija besedila (Text Segmentation and Binarization - TSB) se ukvarja z ločevanjem slikovnih pik besedila od slikovnih pik ozadja. Izhod te stopnje je binarna slika, kjer se črni znaki besedila pojavijo na belem ozadju.
Segmentacija besedila ponavadi vsebuje dva glavna koraka. Prvič, kvaliteta slike je izboljšana z uporabo različnih filtrov v področju slike ali zaradi večkratnih okvirjev vključenih v video področju. Drugič, slikovni elementi besedila v izboljšani sliki so ločeni od slikovnih elementov ozadja s pomočjo statističnih metod ali algoritma mejne vrednosti.
Vhodna vrednost te metode je prvotna slika, koordinate besedila povezujejo polja, ki smo jih dobili že s Sobelovim algoritmom v poglavju 2.1.
Pri segmentaciji in binarizaciji besedila se lahko ločevanja slikovnih elementov besedila od slikovnih elementov ozadja lotimo na dva načina:

- pristop na osnovi mejne vredosti,
- strojno izveden pristop.

Podroben postopek segmentacije in binarizacije je opisan v poglavju. 4.4.3.

4.4.2 Barvni model RGB

RGB je kratica, ki prihaja iz angleščine in pomeni »**red**«, »**green**« in »**blue**«, ki so tudi osnovne barve modela. Vsaka od teh barv se lahko pojavi v 256 odtenkih, kar skupno znaša 16.777.216 barv (256^3).
Barve RGB se pogosto zapisujejo v šestnajstiških trojčkih (predvsem spletne barve v HTML in CSS). Trojček vsebuje tri šestnajstiške vrednosti, ki ponazarjajo količino osnovne barve. Tako na primer FF0000 določa rdečo barvo.



Slika 21: Mešanje barv.

24-bitne barve

V digitalni obliki lahko definiramo barvo s številom med 0 in 255, s kombinacijo teh treh števil pa kombinacijo treh osnovnih barv. Na primer:

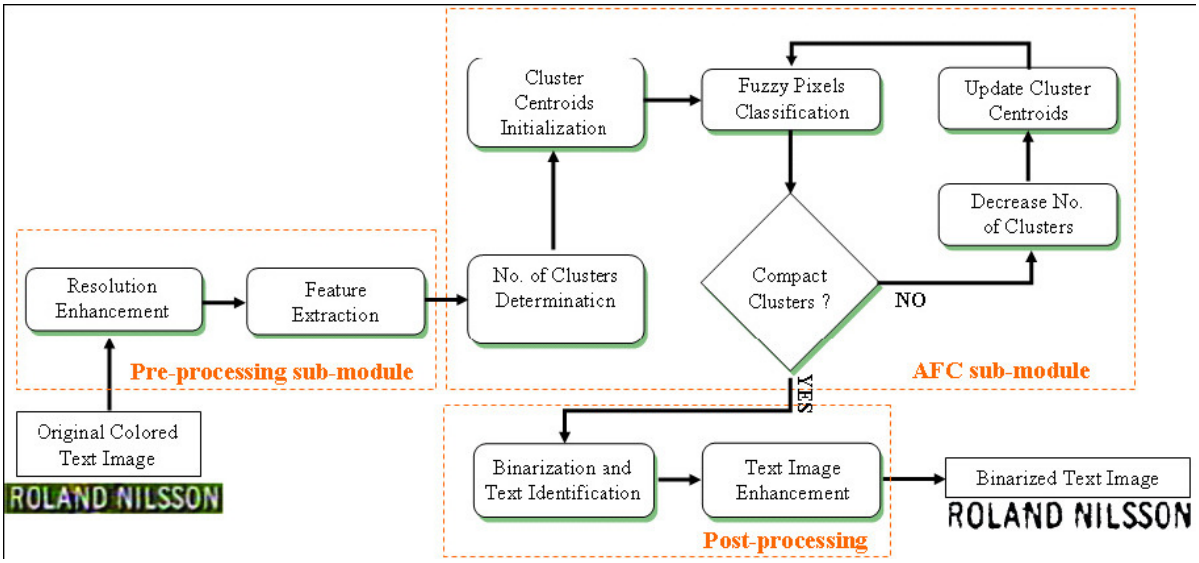
- (0, 0, 0) je črna
- (255, 255, 255) je bela
- (255, 0, 0) je rdeča
- (0, 255, 0) je zelena
- (0, 0, 255) je modra
- (255, 255, 0) je rumena
- (0, 255, 255) je sinja
- (255, 0, 255) je vijolična

4.4.3 Postopek segmentacije za neurejeno prilagodljivo besedilo (Adaptive fuzzy text segmentation method)

AFTS lahko razdelimo na sedem glavnih korakov:

1. Izboljšanje ločljivosti (Resolution enhancement);
2. Izločanje in normalizacija značilk (Feature extraction and normalization);
3. Določitev začetnega števila grozdov (Determination of the initial number of clusters);
4. Določitev začetnega števila centroidov (Determination of the initial clusters centroids);
5. Neurejeno prilagodljivo grozdenje (Adaptive fuzzy clustering - AFC);
6. Binarizacija in identifikacija besedila (Binarization and text identification);

7. Izboljšanje slike besedila (Text image enhancement).

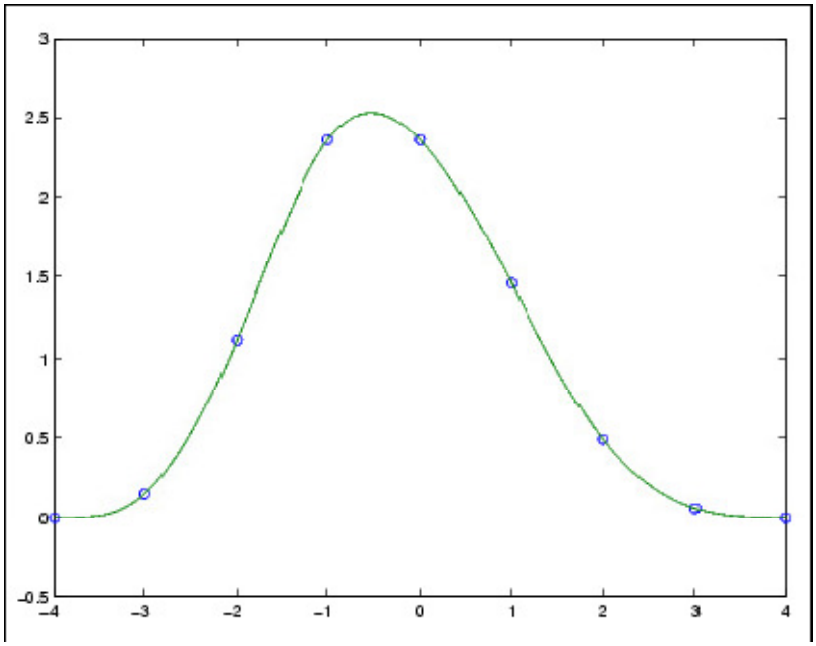


Slika 22: Diagram postopka segmentacije za neurejeno prilagodljivo besedilo

4.4.3.1 Izboljšanje ločljivosti

Video posnetki so ponavadi digitalizirani z nizko ločljivostjo 72 dpi, ki pa ni najbolj primerna za segmentacijo in binarizacijo besedila. Zaradi tega je višina znakov v MPEG-1 video posnetkih manjša od 11 pikslov. Čeprav so takšna besedila še vedno razpoznavna za človeka, pa so za običajni OCR še vedno zelo velik izziv. Standardni OCR sistemi so zasnovani za prepoznavanje besedila v sliki, ki so skenirane z ločljivostjo vsaj 300dpi in višino besedil najmanj 40 slikovnih pik. Metoda segmentacije in nadaljni koraki OCR dajo veliko boljše rezultate na izboljšanih slikah z ločljivostjo 300 dpi, kot pa na enaki sliki z ločljivostjo 72 dpi. Tudi algoritem segmentacije dosega boljše rezultate, če besedilo ni premajhno. Izboljšanje ločljivosti slike na 300 dp je naloga predobdelave, ki jo naredimo z uporabo kubične interpolacije, pri čemer je razmerje še vedno ohranjeno. Kubična interpolacija uporablja okolico 16 x 16 obdanih pikslov računanih točk, da računa njihove vrednosti. Izračun lahko upoštevamo kot konvolucija 16x16 okolice z kubično interpolacijo funkcije. Na sliki 23 je predstavljena enodimenzionalna funkcija kubične

interpolacije, ki deluje kot nizkopasovni filter. Kubična funkcija ima zvezni odvod zvezni



Slika 23: Eno dimenzionalna funkcija kubične interpolacije.

v nasprotju z linearno funkcijo, ki vsebuje nenadne spremembe v naklonu. Kubična funkcija ima tudi negativne režnje. Tako je inerpolirana vrednost kubične funkcije utežena vsota bližnjih vrednosti slikovnih elementov, minus prispevek oddaljenih slikovnih elementovk. Ta lastnost kubične funkcija zmanjša učinek nizkoprepustnega filtriranja in rezultira v boljši ostrini v primerjavi z rezultati, ko uporabljamo linearne funkcije.

4.4.3.2 Izločanje in normalizacija značilk

Za čim boljše klasificiranje slikovnih pik kot besedilo ali ozadje, obravnavamo več barv in teksturnih značilnosti. Osnovne značilnosti barve so sestavljene iz barvnih komponent slikovnih pik. Podatki o barvi so izraženi v barvnem prostoru RGB (poglavje 4.4.2). Vendar pa barvni prostor RGB in njegovi linearni odvodi ne predstavljajo enotnega barvnega prostora. V nasprotju s tem, je CIE L*a*b barvni prostor enotni barvni prostor, kar pomeni, da evklidska razdalja med dvema točkama barve v CIE L*a*b barvnem prostoru ustreza percepcijski razliki med dvema barvama pri človeškem vidu. Zato so barvne značilnosti obravnavane v RGB in CIE barvnem prostoru.

Za preučevanje lastnosti lokalne slike uporabljamo majhna drseča okna m x n (3x3 slikovnih pik), ki se premikajo preko besedila na sliki.
Ta pristop temelji na dveh opažanjih:

- Znaki imajo ponavadi zelo karakteristično strukturo,
- Meja superponiranih znakov rezultira v robove z visokim kontrastom.

Nad sliko lahko uporabimo valčno (Wavelet) transformacijo, ki te lastnosti posreduje naprej algoritmu klasificiranja. Za standardni odklon valčnih koeficientov v drsečem oknu pričakujemo, da bo znotraj teksture znakov nizek, višji pa na njegovih mejah. Tako so v segmentaciji slike meje znaka izboljšane z uporabo teh značilnosti. Ker je pričakovano, da bo kontrast besedila visok glede na ozadje na sivinski preoblikovani sliki (vendar ne v vsakem barvnem kanalu), je ponovno potrebno uporabiti valčno transformacijo na sivinski verziji slike. Na ta način so z upoštevanjem teksture slikovnih pik valčni koeficienti v visokofrekvenčnih pasovih (LH and HL) in njihovi standardni odkloni prav tako vključeni v množico možnih značilnosti. Standardni odklon valčnih koeficientov v drsečem oknu dimenzije mxn, ki so postavljeni na poziciji (x, y), določimo na naslednji način:

$$stDev_{window(x,y)}=\sqrt{\frac{1}{mn}\sum_{i=-\frac{m}{2}}^{\frac{m}{2}}\sum_{j=-\frac{n}{2}}^{\frac{n}{2}}(W(x+i,y+j)-mean_{window}(x,y))^2} \qquad (4.17)$$

Kjer je W (x+i, y+j) vrednost valčnih koeficientov na položaju (x+i, y+j), $mean_{windov}$ (x,y) pa izračunamo z uporabo naslednje enačbe:

$$mean_{window(x,y)}=\frac{1}{mn}\sum_{i=-\frac{m}{2}}^{\frac{m}{2}}\sum_{j=-\frac{n}{2}}^{\frac{n}{2}}(W(x+i,y+j) \qquad (4.18)$$

Na koncu so vse izločene značilnosti komponent normalizirane na območje [0, 1].

4.4.3.3 Določitev začetnega števila grozdov

Pri tem postopku število grozdov definiramo v času delovanja in se spreminja v skladu s sliko, ki se segmentira. Pristop temelji na dejstvu, da je ozadje za besedilom pogosto

večbarvno in ima zato različne teksturne lastnosti od slike do slike. Število grozdov je definirano s pomočjo uporabe heurističnega postopka, ki ga sestavljata naslednja dva koraka.

- **1. Korak:** Iz vhodne slike izberemo določeno število pomembnih pikslov (npr. piksli se nahajajo na sredinski vrstici).

- **2. Korak:** Izbrani piksle se razdelimo v množice pikslov s podobnimi barvami. Primerna barvna razdalja je uporabljana za merjenje podobnosti med dvema različnima barvama. Če je razlika med barvama dveh pikslov pod mejno vrednostjo ($th_{similar}$), potem sta si piksla med seboj podobna. Rezultat te stopnje so skupine pikslov s podobnimi barvami.

Predpostavljamo, da je število ustvarjenih množic barv posredni pokazatelj možnega števila barv v sliki. Zato je to število uporabljeno kot začetno število grozdov C pri mehkem (fuzzy) »C-means« algoritmu.

4.4.3.4 Inicializacija centroidov grozda

Korak inicializacije je pomemben, ker lahko različne izbire začetnih centroidov grozdov vodijo potencialno do različnih particij, ki se potem predstavijo kot različni segmentacijski rezultati. Predlagana inicializacijska metoda uporablja skupine barv, definirane že v prejšnjem koraku. Poteka pa na sledeč način:

1. Korak: Predstavniško barvo izračunamo za vsako skupino kot srednjo barvo vseh barv, ki pripadajo tej skupini. Predpostavljamo, da ocenjene barve C_j za j=1.. C sestavljajo predstavniške barve vhodne slike in se uporabijo kot izhodišče za začetno mehko (fuzzy) delitev v naslednjem koraku.

2. Korak: Funkcija, ki odloča o pripadnosti barve u_{ij} je definirana z uporabo enačbe 4.19 [1]

$$u_{ij} = \begin{cases} 1.0 & \text{if } \delta(x_i, c_j) = 0\ , \\ 0.0 & \exists\ k \neq j \text{ fulfilling } \delta(x_i, c_k) = 0, \\ (\sum_{k=1}^{C} (\frac{\delta(x_i, c_j)}{\delta(x_i, c_k)})^{\lambda})^{-1} & \text{otherwise} \end{cases} \tag{4.19}$$

Pri tem je λ utežen parameter, ki ga uporabljamo kot merilo za pripadnost x_i v c_j, ponavadi z isto vrednostjo kot je vrednost parametra algoritma FCM m (m= λ =2). $\delta(x_i,c_j)$ pa je evklidna razdalja med barvama x_i in c_j.

4.4.3.5 Neurejeno prilagodljivo grozdenje

Algoritem FCM uporabljamo za klasificiranje pikslov slike besedila v C grozdu, kjer C in začetno matriko pripadnosti $\underline{u}$ določimo kot v predhodnih korakih. Evklidsko razdaljo uporabimo za merjenje podobnosti med lastnostmi, ki predstavljajo vsak piksel. Po konvergenci procesa grozdenja algoritem nadaljuje z naslednjimi koraki:

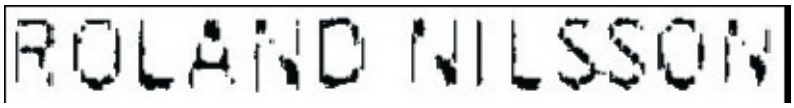
1. Korak: «mehka« pripadnost se pretvori v »fiksno« pripadnost na osnovi kriterija maksimalnosti: vsaki piksel klasificiramo tako, da pripada grozdu, za katerega ima maksimalno vrednost pripadnosti.

2. Korak: Poprečje (poprečje j) stopnje pripadnosti u_{ij} predmeta x_i v vsakem grozdu j, izvedemo z naslednjo enačbo:

$$mean_j = \frac{\sum_i u_{ij}}{|cluster_j|}, \forall i | x_i \in cluster_j \tag{4.20}$$

Glede na to, da stopnja pripadnosti u_{ij} podaja mero podobnosti med točkama x_i in centrom grozda j, bo vrednost povprečja *j* podajala približno informacijo o tem, kako kompakten je grozd j.

a) Vhodna slika.



b) Segmentacija slike brez AFC; na koncu ostane 6 grozdov, C=6.

c) Segmentacija slike z AFC; na koncu ostanejo 4 grozdi C=4.

Slika 24: Vpliv AFC na končni rezultat segmentacije besedila.

3. Korak: če pogoj z enačbo:

$$mean_j >= th_{compact}, \forall cluster_j \tag{4.21}$$

ni izpolnjen, potem se število grozdov dekrementira: C=C-1. Dva grozda z najnižjo povprečno vrednostjo se združi v enega, in izračuna se nov center kot povprečje starih centrov grozdov. Prav tako lahko upoštevamo tudi ostale kriterije, npr. združevanje grozdov z najnižjo povprečno vrednostjo *j* z njihovimi najbližjimi grozdi barvnega prostora. Po koncu inicializacije z novimi centri, se piksli ponovno klasificirajo. Postopek klasificiranja se ponavlja vse dokler ne izpolnimo zgoraj navedenega pogoja (4.21). Na ta način preprečimo prekomerno segmentacijo.

Na sliki 24 je primer, kako je segmentacijski rezultat izboljšan z uporabo predstavljenega algoritma AFC namesto enostavnega FCM.

4.4.3.6 Binarizacija in identifikacija besedila

Po izvedbi zgoraj razloženih korakov, ustvarimo C grozdov pikslov. Za vsak grozd j, generiramo binarno sliko $bImage_j$, ki ima enake dimenzije, kot originalna slika z

besedilom. Le-ta označi vse tiste piksle, ki pripadajo grozdu j s črno barvo, ostale pa z belo.

Nadalje moramo poiskati ustrezen grozd. To je tisti, ki vsebuje piksle besedila. Za ta namen lahko uporabimo algoritem vrednotenja (glej algoritem 14) [1]. Le-ta izkorišča dejstvo, da izkazujejo povezane komponente in njihovi omejujoči pravokotniki (slika 24) v

a) Piksli klasificirani v prvi grozd.

b) _Piksli klasificirani v drugi grozd.

c) Piksli klasificirani v tretji grozd.

Slika 25: Predstavljeni pravokotnik omejuje izločene povezane komponente za vsak kandidat besedila v sliki (C = 3)

binarni sliki besedila podobne prostorske in geometrične lastnosti kot ostale slike kandidatov. Razlog za to je, da so znaki v besedilu pogosto vodoravno poravnani oz. ležijo v ravni črti. Tako lahko pričakujemo, da bodo y koordinate spodnjih desnih kotov omejujočih pravokotnikov imele zelo podobne vrednosti, če jih bomo ocenjevali v pravilnih binarnih slikah besedil. Posledično bo njihov standardni odklon (stDevYCCj) blizu vrednosti 0. Prav tako lahko pridemo do ugotovitve, da omejujoči pravokotniki komponent slike besedila kažejo na podobna razmerja glede višine in širine. Po izločanju vseh povezanih komponent in ocenitvi omejujočih pravokotnikov v vsaki sliki $bImage_j$, ocenimo različne značilnosti, ki temeljijo na zgoraj omenjenih opažanjih oziroma dejstvih. Algoritem vrednotenja deluje na naslednji način. Najprej vse kandidate slik, ki ne izpolnijo kriterija minimuma npr.: (($densityBlank_j > 0$) in (($densityBlank_j < 0.7$), izvzamemo iz nadaljnjega vrednotenja. Nato pričnemo izvajati proces vrednotenja, ki se sestoji iz več korakov, pri čemer vsakega od preostalih kandidatov vrednotimo na podlagi vrednosti

vsake ocenjene značilnosti. Maksimalna vrednost za vrednotenje značilnosti je Max = C - 1, pri tem C označuje število slik kandidata. Znotraj koraka vrednotenja dodelimo kandidatu z najboljšo vrednostjo značilnosti maksimalno oceno značilnosti, drugemu najboljšemu kandidatu dodelimo oceno max – 1, itd.

Algorithm 14: The pseudocode of the text cluster identification algorithm

Input: The clusters created by the AFC

Output: The binarized text image: $textImg_{M \times N}$

For each $cluster_j$ create a binary image: $bImage_j$;
foreach $bImage_j$ **do**
    Extract the connected components: $cc_j[]$;
    Estimate the corresponding bounding rectangles: $bRect_j[]$;
end
for $j = 1$ *to* C **do**
    $stDevHCC_j$ = StandardDeviation(Height($bRect_j[]$));
    $stDevWCC_j$ = StandardDeviation(Width($bRect_j[]$));
    $stDevSizeCC_j$ = StandardDeviation(Size($bRect_j[]$);
    $stDevYCC_j$ = StDevY-coordLowerCorner($bRect_j[]$));
    $mean_j$ = MeanClusterMembership($cluster_j$);
    $densityBlank_j$ = Density(VerticalProjection($bImage_j$));
    ...;
end
for $j = 1$ *to* C **do**
    if $(densityBlank_j > 0)$ *AND* $(densityBlank_j < 0.7)$ **then**
        Rate $bImage_j$ based on the evaluated features;
    end
end
$textImg$ = HighestRate($bImage_j$);

Slika 26: Psevdo koda algoritma za klasificiranje slik besedila.

Na primer, za značilnost stDevYCCj, izvedemo ustrezni korak algoritma vrednotenja na naslednji način: sliki, ki ima najnižji standardni odklon *y* pripišemo maksimalno oceno značilnosti in sliki, ki ima najvišji standardni odklon y, pripišemo minimalno oceno

značilnosti. Ko ocenimo vse slike $bImage_j$ za vsako njihovo značilnost, izberemo sliko s skupno najvišjo oceno vrednotenja kot pravilno binarno sliko besedila.

a) Vhodna slika.

b) Segmentacija slike pred postopkom izboljšanja.

a) Segmentacija slike po uporabi postopka izboljšanja.

Slika 27: Vpliv postopkov izboljšanja na končni rezultat procesa segmentacije slike besedila.

4.4.3.7 Izboljšanje slike besedila

Postopek izboljšanja slike besedila izvajamo na binarizirani sliki besedila. Vse povezane komponente, ki so bodisi premajhne ali pa ležijo na meji slike besedila in imajo njihovi omejujoči pravokotniki presek z ostalimi omejujočimi pravokotniki, odstranimo. Tiste komponente, ki ležijo na mejah slike besedila, njihovi omejujoči pravokotniki pa nimajo preseka z ostalimi omejujočimi pravokotniki, samo deloma odstranimo. Uporaba morfološke operacije »open« za ločevanje možnih mej med znaki zaključi proces. Rezultat procesa izboljševanja je prikazan na sliki 27.

4.5 Izločanje značilk (feature extractor)

Po postopku segmentacije in binarizacije imamo pripravljen znak za vhod v naslednji korak, ki znak definira, določi lastnosti, oziroma značilke znaka (*angl. feature extractor*) in

mu priredi vrednosti v obliki vektorja značilk, ki so odvisne od same metode razpoznavanja. Prav definicija značilk je bistvenega pomena za uspeh metode, seveda pa je tudi tu velika odvisnost od vrste pisave oziroma znaka. Rezultat postopka izločanja značilk je vektor značilk, ki vsebuje informacije o znaku, ki predstavljajo vhodni podatek razvrščevalnika. V poglavju 4.4.3.2 smo podrobneje opisali postopek izločanja značilk [5].

4.6 Tehnologija optičnega prepoznavanja znakov (OCR)

Prepoznavanje znakov (Character Recognition – CR) je zadnji korak v procesu razpoznavanja besedila v sliki ali videu. Izvaja optično razpoznavo znakov (OCR) na binarnem slikovnem besedilu in preoblikuje binarno besedilo slike v pripadajočo ASCII besedilo. Znak in stopnja prepoznavnosti besede je definirana s spodnjo enačbo:

$$CRR(WRR) = \frac{pravilno\ prepoznan\ znak\ (beseda)}{skupno\ število\ znakov\ (beseda)} * 100 \tag{4.22}$$

4.6.1 Zgodovinski razvoj

O zametkih sistemov OCR lahko govorimo že v 19. stoletju, o uporabnih modernejših sistemih pa od leta 1950 dalje. Prvi uspešni poskus na tem področju je izvedel ruski znanstvenik Tyurin leta 1900. Goldberger je leta 1912 patentiral stroj, ki je bral tipkane črke in jih pretvoril v telegrafsko kodo. Leta 1929 je nemški znanstvenik Tausheck prvi patentiral OCR v Nemčiji, štiri leta kasneje pa še Handel v ZDA. To je vzbudilo sanje o stroju, ki bi znal brati črke in številke. Sanje so se začele uresničevati s pričetkom informacijske dobe v 50-ih letih.

Kasnejši najbolj znani poskusi na področju OCR sistemov so Sheppardov bralno pisalni robot imenovan GISMO, Kelner in Glaubermanov magnetni pomikalni register iz leta 1956 itd. Do zgodnjih sedemdesetih let so sistemi OCR podpirali le omejeno število točno definiranih pisav in bili prisotni le v nekaj ozkih, speciliziranih področjih vladnih agencij in velikih korporacij.

Kasnejša predstavitev sistemov, ki so se lahko naučili razpoznavanja različnih pisav, je dodatno razširila trg in rezultirala v še večji priljubljenosti ter komercialni uspešnosti sistemov OCR. Še posebej je vredno omeniti Hitachijev H 8959 laserski skener z OCR

procesorjem v sredini 70-ih, ki ga imajo za enega mejnikov v razvoju OCR-a, saj je imel za tisti čas izredne sposobnosti in relativno nizko ceno. Z naglim razvojem strojne opreme so se sistemi OCR preselili v območje programskih in ne več strojnih rešitev. Topološke oziroma strukturne analize so izpodrinile prepoznavanja s pomočjo primerjave in so teoretično omogočile prepoznavanje različnih znakov (pisave) glede na unikatne lastnosti njihove oblike. Danes pa vse bolj prevladujejo algoritmi na osnovi tehnologije nevronskih mrež [5].

4.6.2 Delitev sistemov za prepoznavanja znakov

Sisteme OCR lahko delimo na več načinov. Opisali bomo delitev glede na namen in glede na vrsto zapisa. Metode in tehnologije OCR bomo predstavili v nadaljevanju tega poglavja.

4.6.2.1 Delitev glede na namen

Delitev sistemov na industrijske in neindustrijske sisteme OCR lahko jemljemo kot delitev glede na namen. Razlike so predvsem v robustnosti in zanesljivosti sistemov. Pri industrijskih sistemih so napake nedopustne in v primeru, da verjetnost prepoznanega ni zelo velika, je potrebno sporočiti napako; medtem ko je pri neindustrijskem OCR-u rezultat najbolj verjeten znak in redkeje prihaja do zavrnitve znaka.
Prav tako v industrijskih sistemih OCR običajno razvijemo algoritme in metode prepoznavanja za vsak problem posebej, predvsem zaradi prilagoditve naboru znakov in pogojem delovanja [5].

4.6.2.2 Delitev glede na vrsto zapisa

Optično prepoznavanje znakov delimo glede na vrsto zapisa na:

- Prepoznavanje ene vrste pisave (angl. **Fixed-font** character recognition), ki je vnaprej znana, kar je primer predvsem pri industrijskih sistemih. Prevladujejo primerjalne metode.

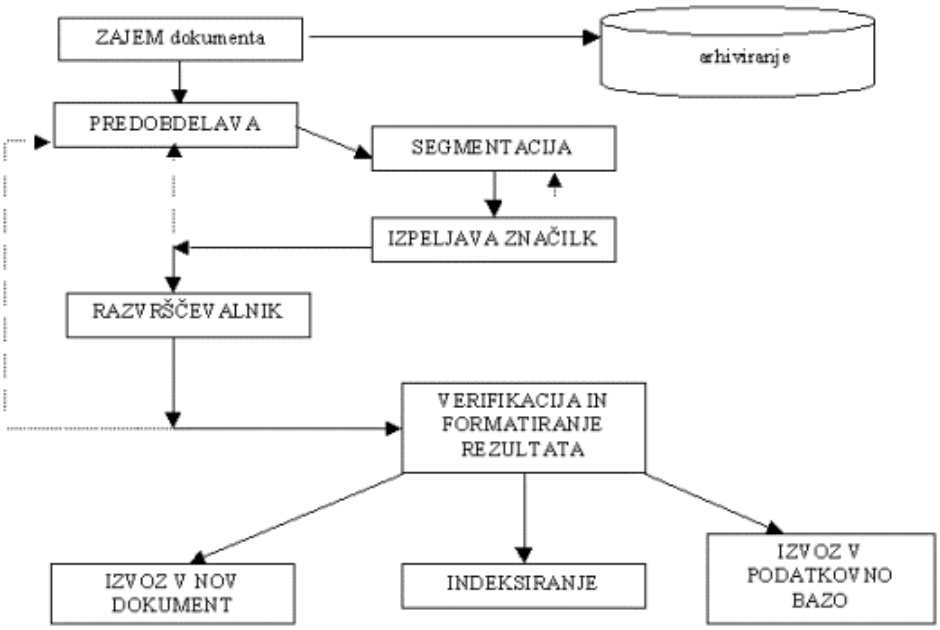
- Prepoznavanje poljubne pisave (angl. **Omnifont** character recognition), ki zajema prepoznavanje natipkanega besedila in naj bi bilo čim bolj neodvisno od pisave. Omogočalo naj bi tudi interakcije s samim uporabnikom, kar zajema tudi učenje znakov in stilov pisav. Sodobni sistemi imajo povezavo z raznimi slovarji, ki omogočajo avtomatično odpravljanje napak. Prevladujejo metode strukture oziroma topološke analize.

- **ICR** (angl. Intelligent character recognition) – inteligentno prepoznavanje znakov. Ta oblika se navezuje predvsem na prepoznavanje ročne pisave. Prevladujejo metode na osnovi nevronskih mrež.

- **OMR** (angl. Optical mark recognition) – prepoznavanje označevanj, kjer gre za prepoznavanje označevanj na obrazcih; izhodni rezultat je označeno/neoznačeno.
- Prepoznavanje črtnih kod (angl. **Barcode** recognition)

4.6.3 Postopek prepoznavanja znaka

Pri obravnavanju zgodovinskega razvoja smo že omenili, da poznamo dve veliki skupini metod za prepoznavanje: **matrično primerjalno** in **strukturno razvrščevalno metodo.** Znane so še metode na osnovi verjetnostne teorije in pa seveda metode na osnovi **nevronskih mrež**. Velikokrat se v sistemih metode med seboj prepletajo in tako dobimo različne hibridne strukture.



Slika 28: Osnovni koraki OCR razpoznavanja [5].

4.6.3.1 Pred obdelava slike

V želji po doseganju čim boljšega rezultata OCR razpoznavanja, je smiselno slikovni format nekoliko transformirati pred samim procesom OCR razpoznavanja. Lahko govorimo o prvem delu razpoznavanja, v katerega lahko uvrstimo detekcijo in lokalizacijo, segmentacijo in binarizacijo besedila. Procese, ki smo jih opisali že v prejšnjih poglavjih.

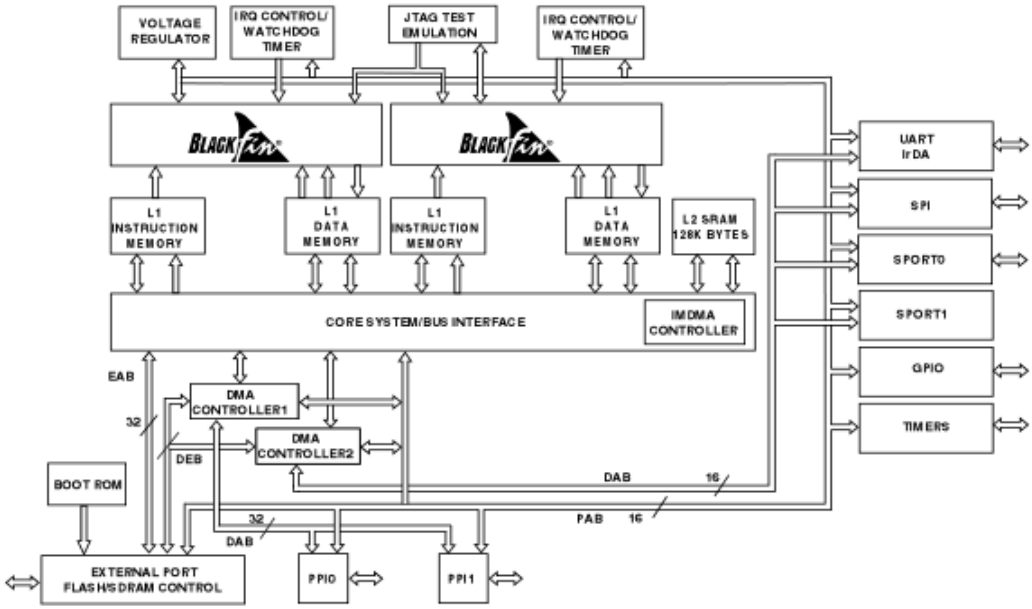
4.6.3.2 Končna obdelava

Poleg človekovega preverjanja rezultata razpoznavanja, vsebujejo sistemi tudi različne metode avtomatskega popravljanja, oziroma verifikacije rezultata. Prvi korak je preverjanje obstoja besede v slovarju (*angl. spell-checking*), ki je v nekaterih sistemih vključena tudi v detekcijo hiper ali podsegmentacije znaka. Nadalje lahko verifikacija poteka na osnovi sistema različnih pravil. Ta pravila definirajo bolj in manj verjetna zaporedja znakov in ostale lastnosti jezika, kot je na primer v angleščini velika verjetnost črke “a”, če se znak pojavi samostojno. Seveda se tovrstne tehnike uporabljajo le v primeru nezanesljivo razpoznanega znaka. Zelo uporabno je na primer barvno označevanje “sumljivo” razpoznanih znakov, kjer lahko tudi barvo vežemo na verjetnost napačnega

razpoznavanja. Po verifikaciji rezultata je potrebno rezultat še primerno oblikovati, glede na strukturo originala in nastavitve uporabnika.

5 DSP RAZVOJNI SISTEM BLACKFIN

Blackfin je družina 16 oziroma 32 bitnih mikroprocesorjev, ki jih je razvilo podjetje Analog Devices. Za to družino je značilno, da imajo vgrajen digitalni signalni procesor (DSP). Rezultat tega je, da imajo majhno porabo moči in poenoteno procesorsko enoto, ki lahko zažene operacijski sistem, medtem ko se signalni procesor ukvarja s kompleksnimi numeričnimi metodami, kot je video kodiranje v realnem času.

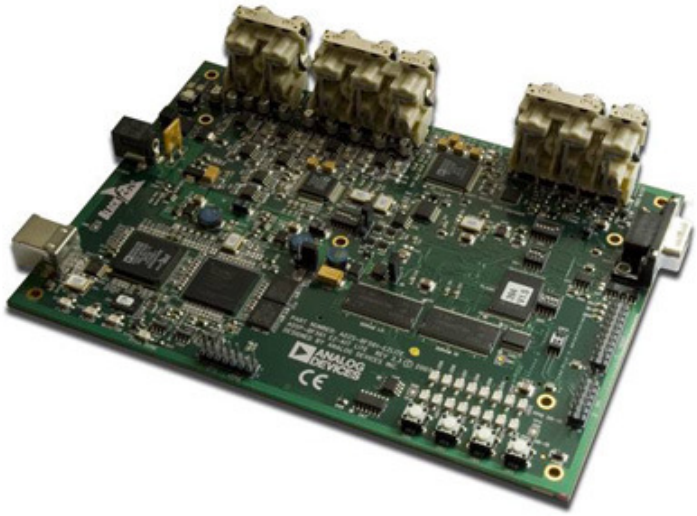


Slika 29: Blokovna shema razvojnega okolja Blackfin.

Dvojedrni BF561 procesor, ki je bil uporabljen v našem primeru uporablja:

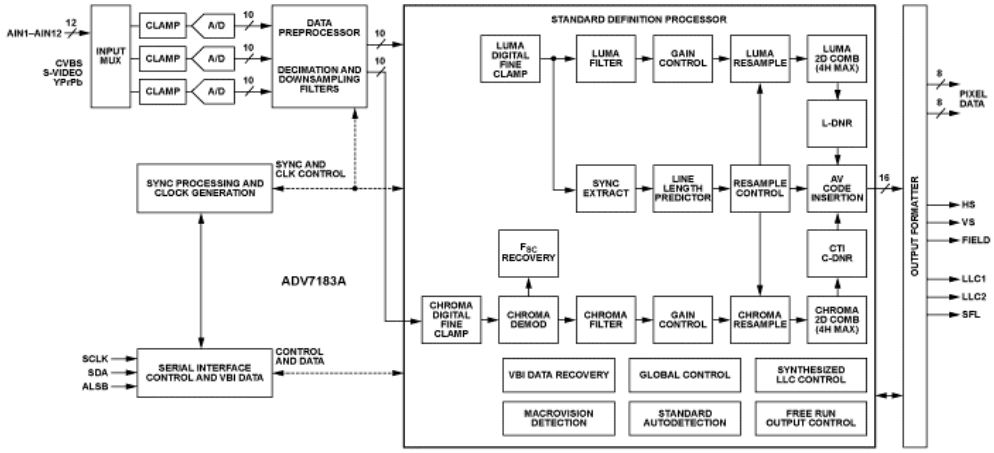
- 600MHz CPU
- 120MHz sistemska ura
- 32-bit zunanje vodilo
- 64MB SDRAM
- 8MB NOR Flash

- AD1836A 96KHz avdio kodek
- 4 RCA avdio vhod (2 stereo)
- 6 RCA avdio izhod (3 stereo)
- ADV7183 video dekodirnik
- 3 RCA video vhodi
- ADV7171 kodirnik
- 3 RCA video izhodi
- UART, LEDs, Push buttons



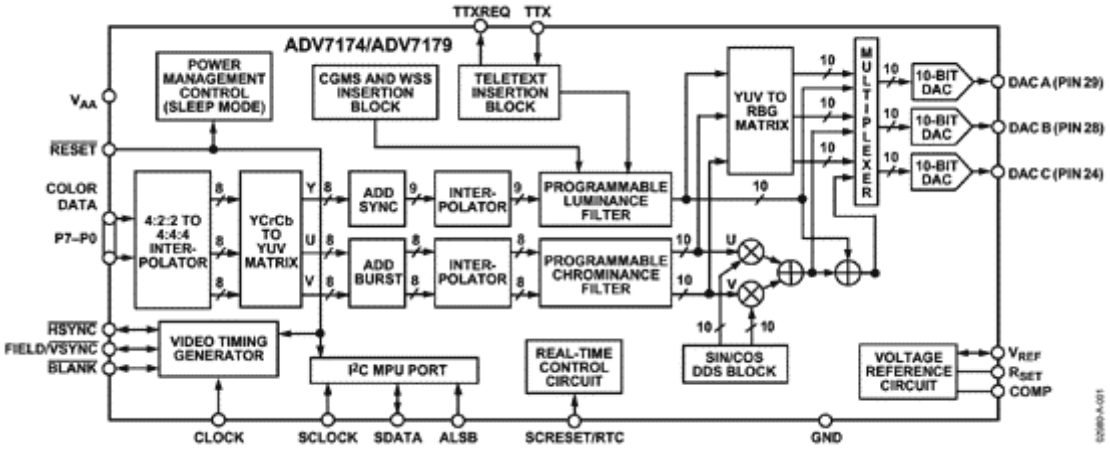
Slika 30: Analog Devices BF561-EZKit.

Video dekodirnik ADV7183B je integriran 10 bitni video dekodirnik, ki samodejno prepozna in pretvori analogni signal s standardi NTSC in PAL v 4:2:2 podatkovni video.



slika 31: Video dekodirnik ADV7183B

Video kodirnik ADV7179 je primeren za široko paleto aplikacij, kot so tretja generacija mobilnih telefonov ali pa se uporabljajo za video izhode pri digitalnih fotoaparatih. V našem primeru je video kodirnik konfiguriran tako, da prikaže na izhodu NTSC ali PAL format.

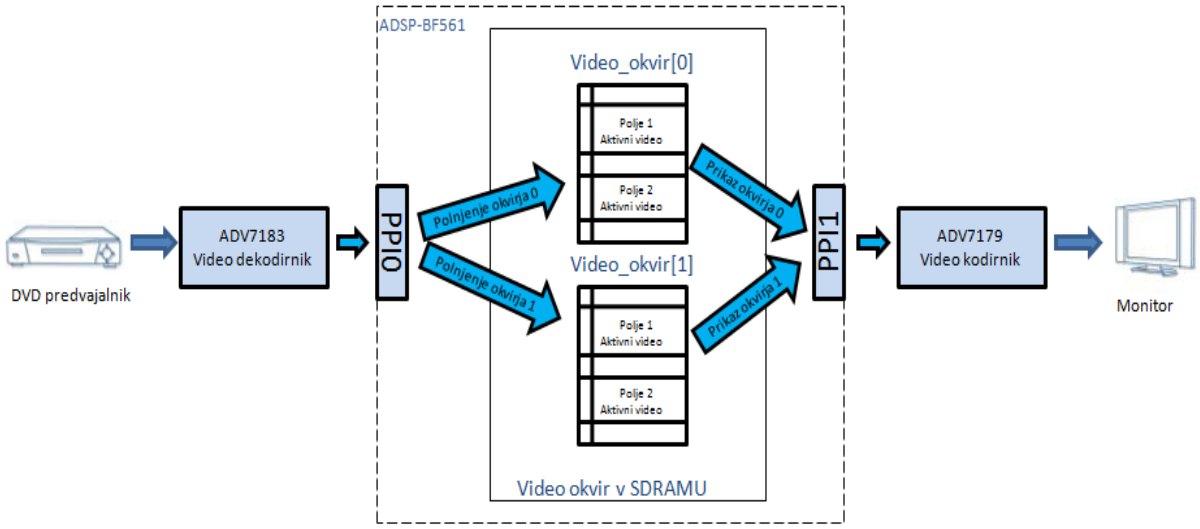


slika 32: Video kodirnik ADV7179.

6 IMPLEMENTACIJA SISTEMA ZA IZLOČANJA BESEDILA V VIDEO SIGNALU

V tem poglavju bomo predstavili glavne dele strojne opremo in opisali samo delovanje programske kode modula, ki izvaja prve korake v kompleksnem procesu prepoznavanja besedila v digitalnem video signalu. Opisali bomo tudi postopek postavitve danega sistema.

6.1 Postopek procesiranja video signala



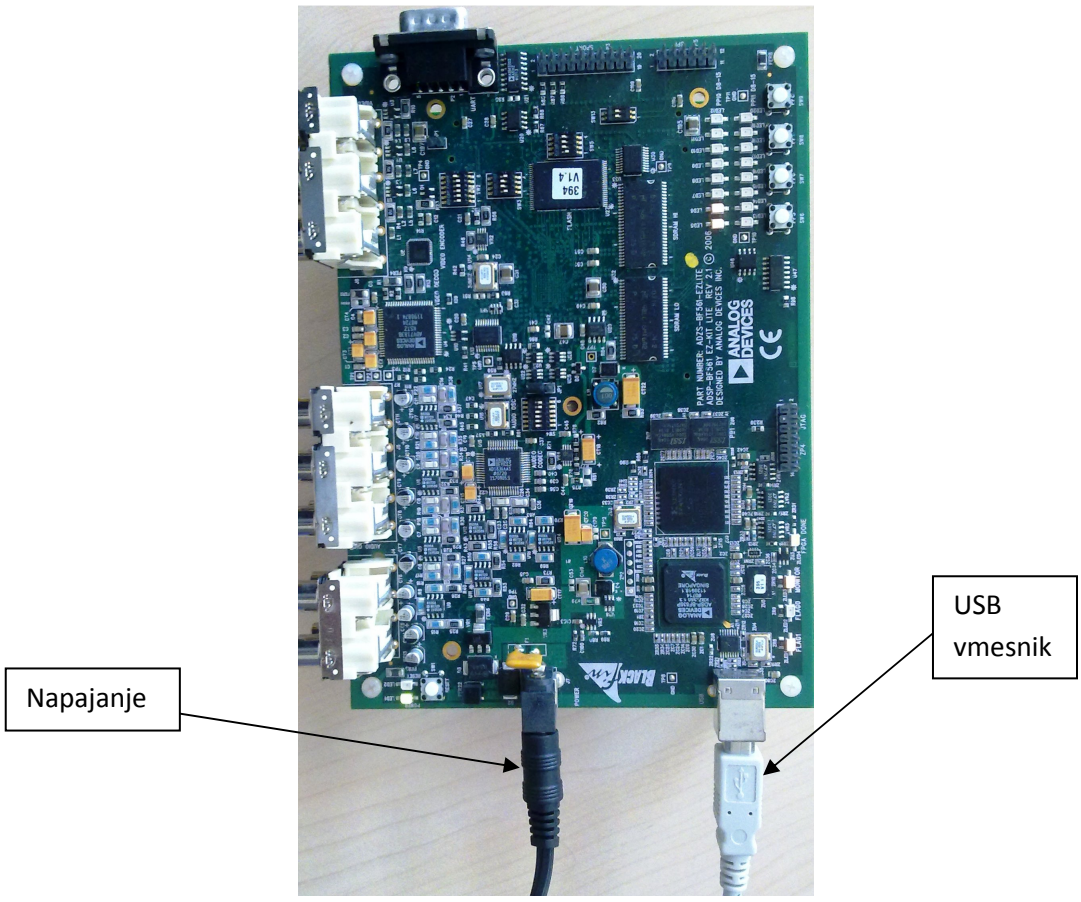
Slika 33: Postopek procesiranja video signala.

Video signal pripeljemo direktno iz DVD predvajalnika v video dekodirnik ADV7183 in nato v SDRAM (slika 33). Video dekodirnik se nahaja na plošči ADSP-BF561 EZ-Kit in je nastavljen tako, da sprejema vhodni vir NTSC ali PAL. Dvojni vmesni pomnilniki so koristni za video aplikacije zato, da lahko imamo v posameznih okvirjih več podatkov, medtem, ko se obdelujejo in prikazujejo predhodni okvirji. SDRAM sestavljajo posamezne

pomnilniške banke, v vsaki banki pa je tudi vmesni pomnilnik. V našem primeru pripeljemo video okvir v banko 0 in banko 1, izhod video signala pa je v banki 2. Algoritem Sobel, s katerim izvajamo zaznavanje robov v video signalu, se za video vhod nahaja v vmesnem polnilniku 0, za video izhod pa v vmesnem pomnilniku 1.
Ko se procesiranje izvede, se izhodni signal iz ADV7179 video kodirnika prenese preko zunanjega TV tunerja AverMedia AVerTV Hybrid Ultra USB na računalnik.

6.2 Postavitev strojne opreme za procesiranja video signala

Najprej povežemo razvojno ploščo Blackfin preko USB vmesnika na računalnik in na napajanje (slika 34).



Slika 34: Priključitev napajanja in USB vmesnika na BLACKfin DSP platformo.

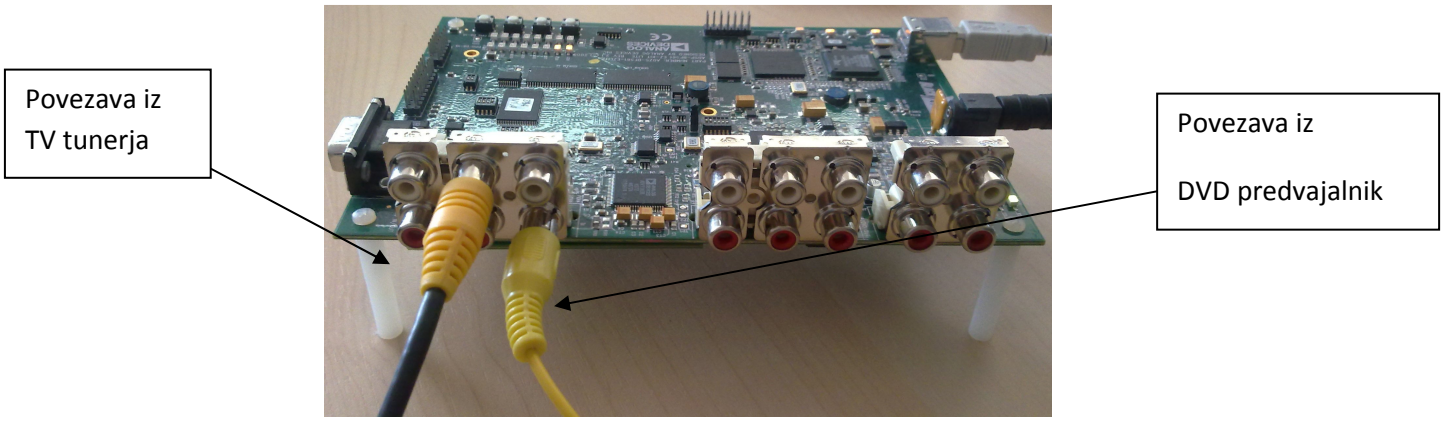
Nato pripeljemo na razvojno ploščo video signal iz DVD predvajalnika (slika 35) s kablom scart-RCA/S-video/DIN (slika 36).

Slika 35: Philips DVDR 75.

Slika 36: Scart-RCA/S-video/DIN.

Scart-RCA/S-video/DIN kabel nastavimo na OUT in ga priključimo na EXT 2 DVD predvajalnik (slika 37), drugi del kabla (rumeni činč – video) pa pripeljemo na razvojno plošč in ga priključimo, kot prikazuje slika 38.

Slika 37: Povezava kabla scart-RCA/S-video/DIN z DVD predvajalnikom.



Slika 38: Povezava na razvojno ploščo BLACKfin BF-561 DSP.

Povezava na zunanji TV tuner je prikazana na sliki 39 če imamo priklopljen zajem na S-video način. Če pa priklopimo zajemanje na »composite« način, pa moramo priključiti TV tuner, kot je prikazano na sliki 40.

Slika 39: TV tuner na S – video način

Slika 40: TV tuner na »composite« način.

Ko vse ustrezno povežemo, se lotimo programskega dela, ki ga predstavljamo v poglavju 6.4.

6.3 Opis strojne opreme

6.3.1 Zunanji TV tuner AverMedia AVerTV Hybrid Ultra USB

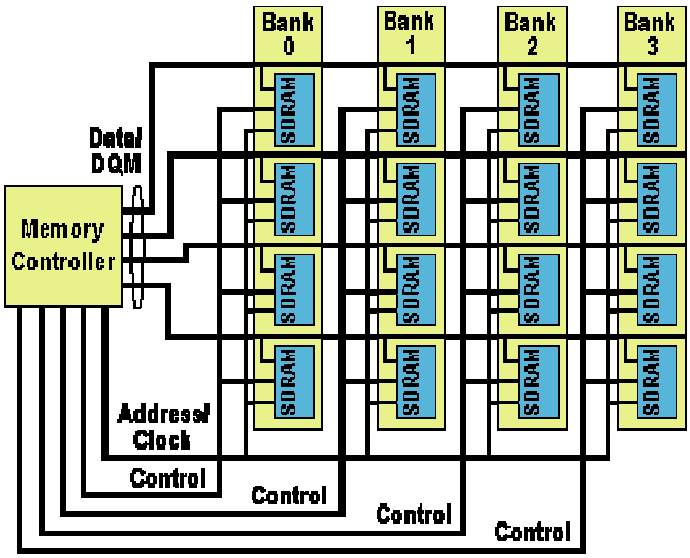
Zunanji TV tuner AverMedia AVerTV Hybrid Ultra USB je naprava, s katero zajemamo preko S-video priključka video signal, ki ga dobimo iz Video kodirnika ADV7179, oziroma ga pripeljemo preko USB vmesnika na računalnik.

Slika 41: Zunanji TV tuner AverMedia AVerTV Hybrid Ultra USB.

6.3.2 Uporaba SDRAMa za delo z video podatki

SDRAM (Synchronous dynamic random access memory) je sinhronski dinamični pomnilnik z naključnim dostopom. Je vrsta DRAMa, ki je narejen za hiter prenos podatkov. Uporablja samodejno naslavljanje.

V SDRAMU imamo več pomnilniških bank, vsak vmesni pomnilnik (buffer) pa mora biti v različni banki. V našem primeru uporabljamo banko 0 in banko 1 za video vhod (video_in_buf 0 in video_in_buf 1), ter banko 2 za video izhod (video_out_buf). Sobel vhod in izhod se nahajata v vmesnem pomnilniku 0 in 1.



Slika 42: Blokovna shema.

6.4 Programski del

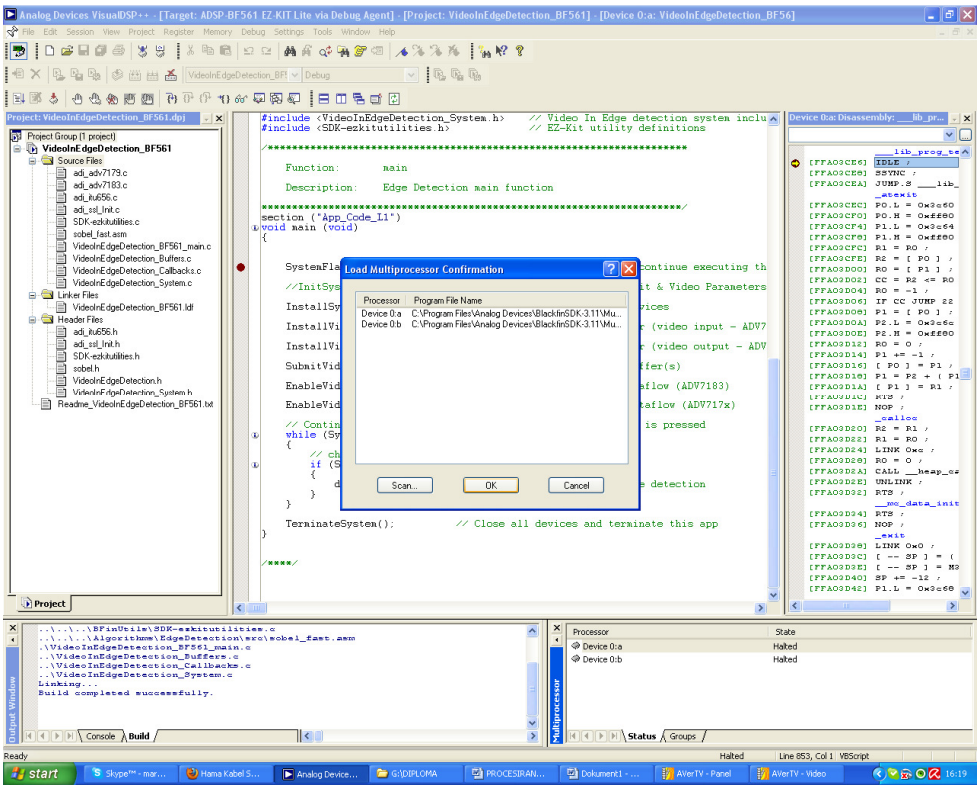
V tem razdelku bomo predstavili programsko okolje Visul DSP 5.0, način nalaganja programa na samo razvojno ploščo BLACKfin BF-561 in izvedli analizo programa za zaznavanje robov v video signalu.

6.4.1 Razvojno orodje Visual DSP 5.0

Razvojno orodje Visual DSP 5.0 je programsko okolje podjetja Analog Devices, ki je namenjeno za programiranje digitalnih signalnih procesorjev (DSP).

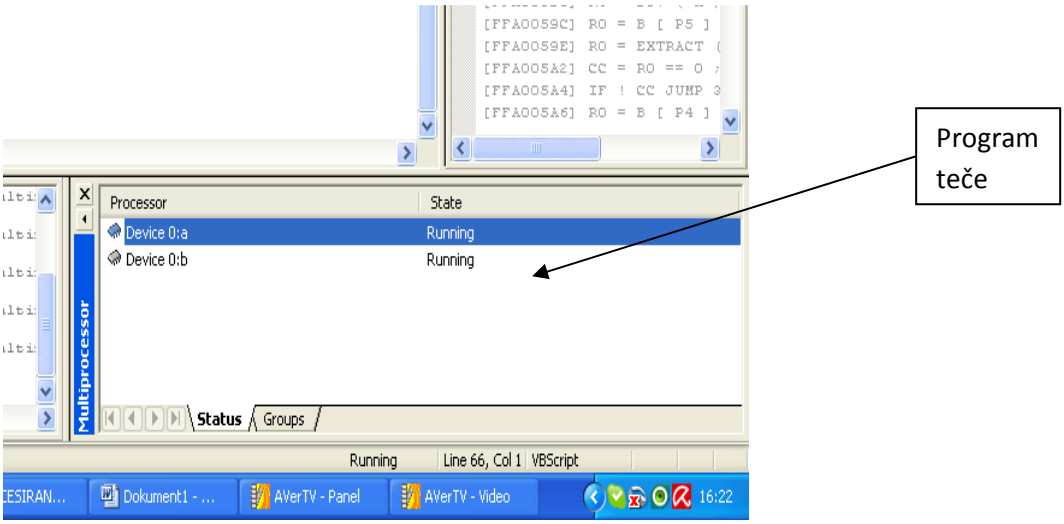
6.4.2 Način nalaganja programa za procesiranje video signala

Ko vzpostavimo delovno okolje (poglavje 6.2) se lahko posvetimo programskemu okolju. Ko program za zaznavanje robov v video signalu prevedemo, se pokaže pogovorno okno (slika 43, v katerem pritisnemo tipko OK. Program se takoj nato naloži na DSP ploščo.



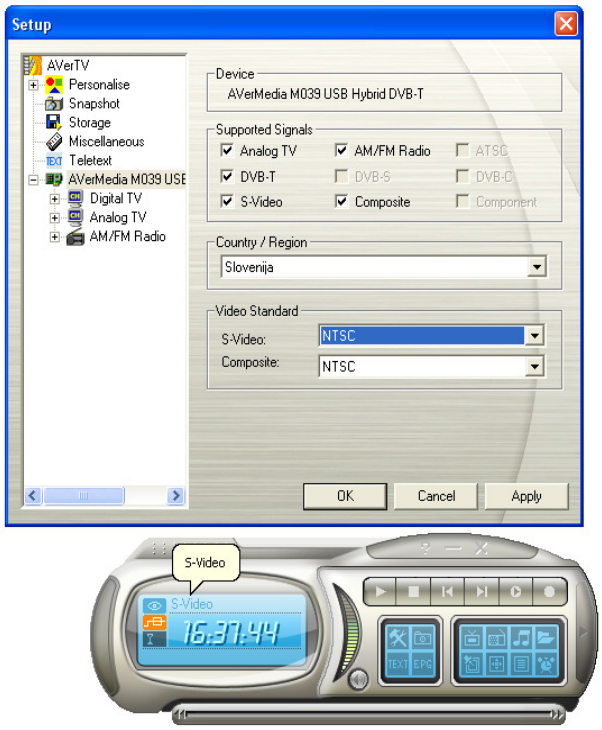
slika 43 : Nalaganje kode v SDRAM.

Ko je program naložen, s tipko (CTRL-F5) poženemo program, kot je prikazano na sliki 44.



Slika 44: Program teče.

Za ogled dobljenega video signala kliknemo ikono programa AVerTV 6 na namizju in izvedemo nastavitve, kot kaže slika 45.



Slika 45: Nastavitev AVerTV 6 za prikaz video signala na monitorju.

6.4.3 Analiza programskega dela modula za zaznavanje robov v video signalu

Osnova modula za detekcijo robov objektov in besedil v digitalnem video signalu je Sobel algoritem, katerega naloga je, da poišče robove na osnovi računanja odvodov in s pomočjo uporabe dveh 3x3 konvolucijskih matrik (poglavje 3.1). V nadaljevanju podrobneje predstavljamo analizo programa danega algoritma v treh blokih, kot je prikazano na slikah 47, 48 in 49. Algoritem teče na DSP platformi Blackfin BF-561, srce algoritma pa je zaradi delovanja v realnem času sprogramirano v strojnem jeziku. Programska koda je napisana v programskem jeziku C, sam projekt pa v programskem okolju VisualDSP 5.0.

6.4.4 Analiza modula za zaznavanje robov v video signalu

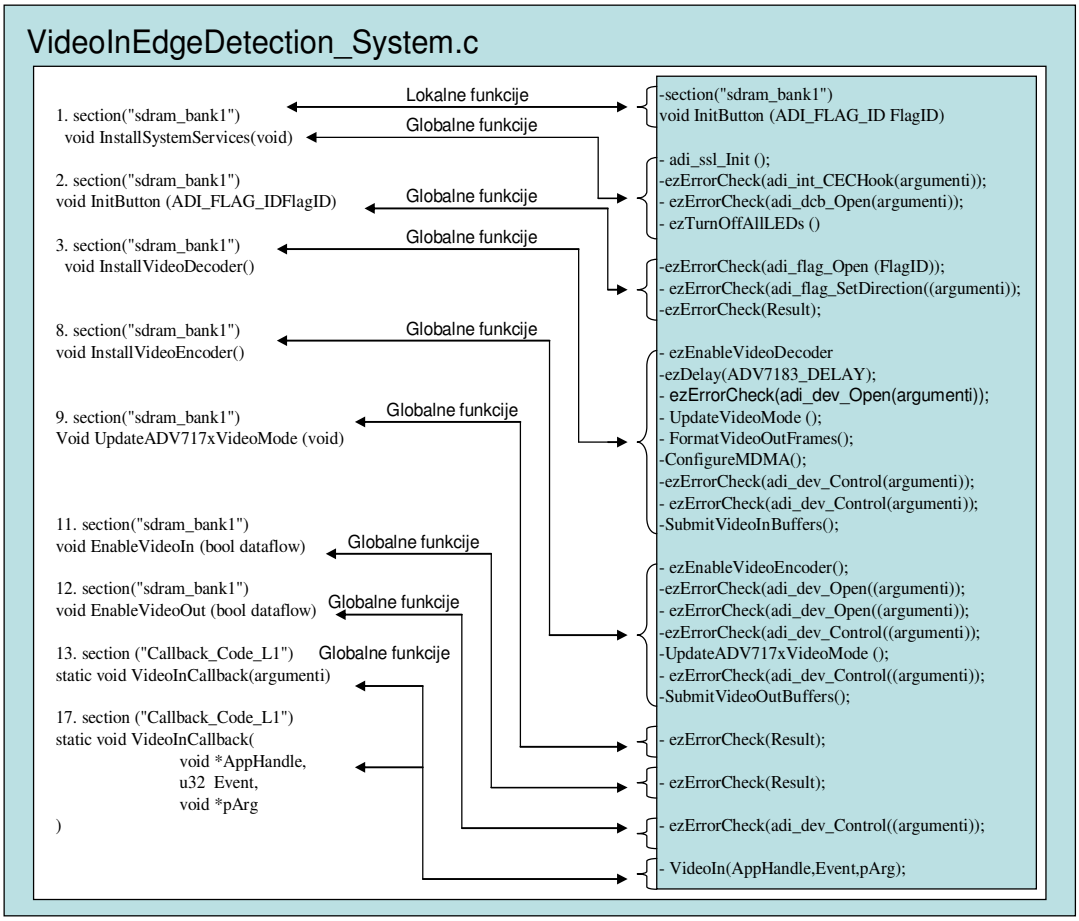
Na sliki 46 je prikazan glavni del programa, v katerem so funkcije, na katerih temelji delovanje zaznavanja robov v digitalnem video signalu.

```
*****************************************************************
Funkcija:     glavna

Opis:    glavna funkcija za zaznavanje robov

*****************************************************************/
section ("App_Code_L1")
void main (void)
{
        printf("main\n");

        SystemFlag.KeepRunningFlag  = TRUE;        // inicializacija zastavice za nadaljevanje izvajanja te
                                                      aplikacije

        //InitSystem ();                           // Inicializacija EZ kit & Video parametrov

        InstallSystemServices ();                  // namestitev sistemskih storitev

        InstallVideoDecoder ();                    // namestitev Video dekodirnika (video vhod -ADV7183)

        InstallVideoEncoder ();                    // namestitev Video kodirnika (video izhod - ADV717x)

        SubmitVideoOutBuffers();                   // izroči video iz vmesnih pomnilnikov

        EnableVideoIn(TRUE);                       // omogoči pretok podatkov za video vhod (ADV7183)

        EnableVideoOut(TRUE);                      // omogoči pretok podatkov za video izhod (ADV717x)

        // nadaljuj izvajanje, dokler prekinitveni gumb (SW9) ni pritisnjen
        while (SystemFlag.KeepRunningFlag == TRUE)
        {
                // preveri, če je Sobel za zaznavanje robov omogočen
                if (SystemFlag.SobelEnable)
                {
                  do_SobelEdgeDetection ();        // če je tako, opravi zaznavanje robov
                }
        }

        TerminateSystem();                         // zapri vse naprave in prekini to
}
```

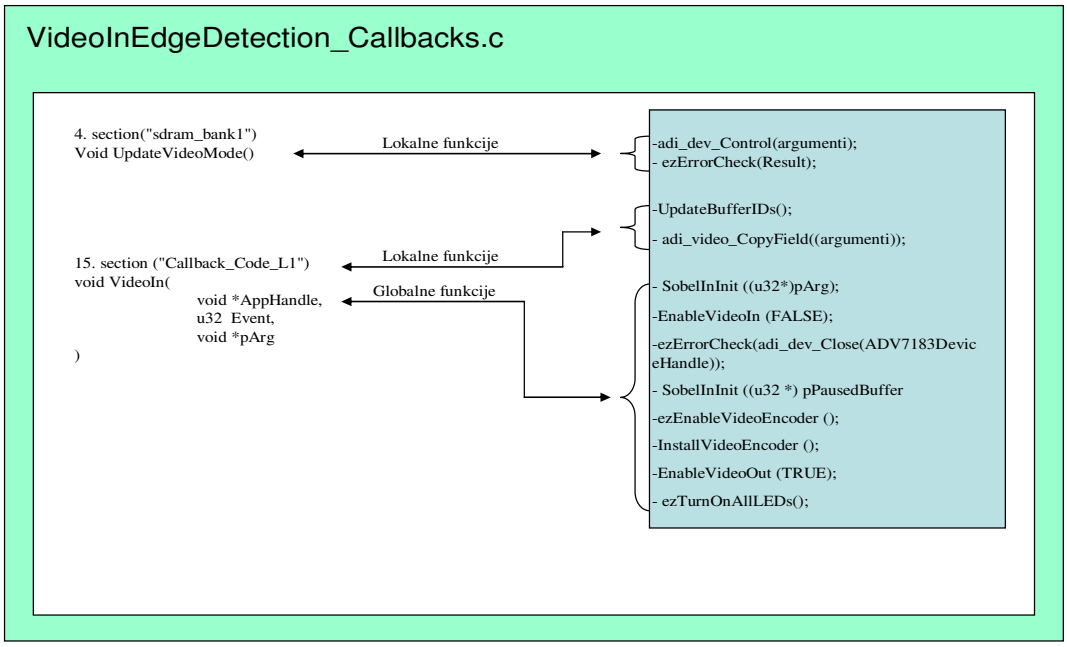
Slika 46: Glavna funkcija »main« programa za zaznavanje robov.

6.4.5 Prikaz poteka klicanja funkcij modula za zaznavanje robov

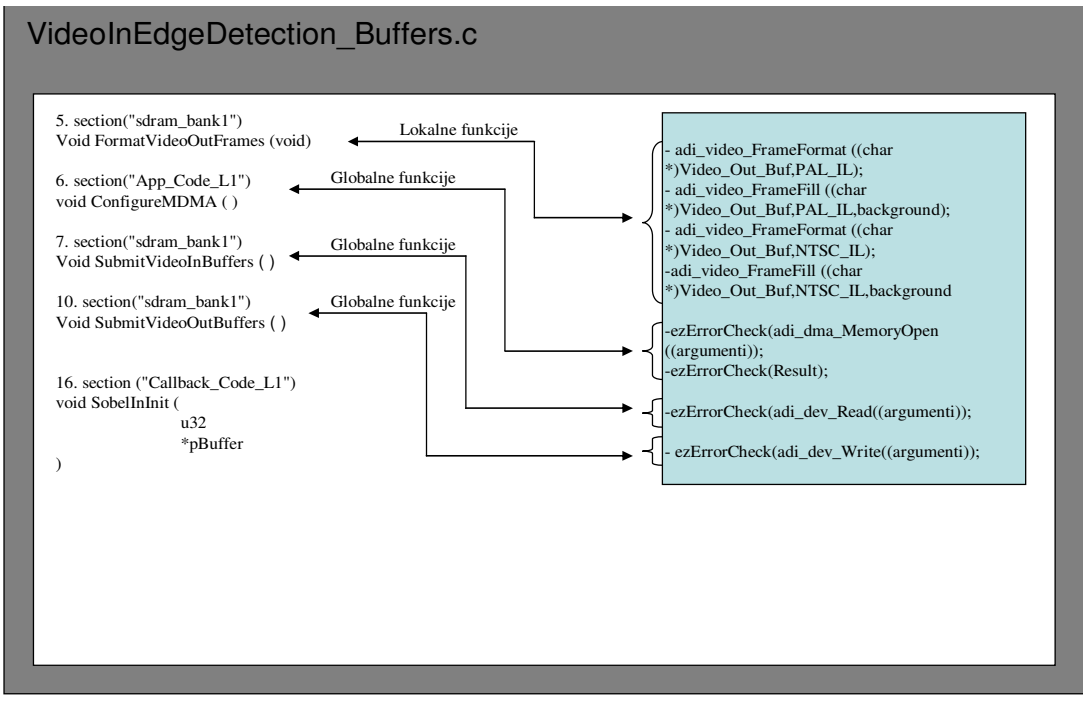
Na spodnjih slikah prikazujemo tri bloke programa, kjer poteka klicanje funkcij, kot je to prikazano s številkami funkcij. Na levi strani vsakega bloka so predstavljeni tudi koraki zaporedja funkcij, ki izvajajo digitalno procesiranje video signala. Na desni strani vsakega bloka pa so predstavljene globalne in lokalne funkcije, ki dopolnjujejo funkcije na levi strani. V poglavju 5.3.1.1 je opisan postopek delovanja programa po korakih.



Slika 47: VideoInEdgeDetection_sistem.



Slika 48: VideoInEdgeDetection_Callbacks.



Slika 49: VideoInEdgeDetection_Buffers.

6.4.5.1 Opis delovanja poteka programske kode za zaznavanje robov

Program se začne izvajati v glavni funkciji (**main** function, slika 46) in najprej kliče funkcijo InstallSystemServices ().

- **Korak 1: InstallSystemServices ();** ta funkcija inicializira EZ kit in delovanje sistema. Inicializira ezKit power, EBIU, DMA..., inicializira zastavice za Ez-Kit gumbe (gumb za prekinitev, gumb za posodobitev video načina, gumb za premor videa, gumb za zajemanje video okvirja, gumb za preklop zaslona).

- **Korak 2: Initbutton();** inicializirane Blackfin zastavice povežemo z EZ-kit gumbi. Nastavimo zastavice in jih nastavimo kot vhod.

- **Korak 3: InstallVideoDecoder();** ta funkcija konfigurira video dekodirnik (ADV7183).
    - Če je definiran ADSPBF535
        - se dekodirnik zaradi stabilizacije zakasni. Nastavimo ADV7183 gonilnike za video vhod in konfiguriramo gonilnike za video dekodirnik.
    - Če definiramo (__ADSPBF561__)
        - Preverimo, če **(ADV717xDeviceHandle==NULL)**
            - Če je tako, osvežimo vhodni video način **UpdateVideoMode (),**
            - Oblikujemo video izhod Sobela v vmesnem pomnilniku za specifičen video način **FormatVideoOutFrames(),**
            - konfiguriramo in namestimo MDMA za odkrivanje robov s Sobelovim algoritmom **ConfigureMDMA()**.
        - V tem koraku tudi preverimo in nastavimo ustrezen video način (ali je ADV7183 v PAL ali v NTSC načinu). Preden pripravimo video vhod vmesnega pomnilnika **SubmitVideoInBuffers()**, še nastavimo metodo pretoka podatkov in omogočimo pretok (streaming).

- **Korak 4: UpdateVideoMode();** ta funkcija izvaja posodobitev in preklapljanje med PAL in NTSC načinom.

- **Korak 5: FormatVideoOutFrames();** ta funkcija preverja oblike vhodnega okvirja (ali PAL ali NTSC). Če je PAL, zapolni video iz okvirja z barvo ozadja, enako velja tudi za NTSC način.

- **Korak 6: ConfigureMDMA();** v tem koraku namestimo in konfiguriramo pomnilnik DMA.
  - konfiguracija dma 2D strukture,
  - video vhodni tok podatkov za Sobel algoritem, ki je v vmesnem pomnilniku,
  - tok podatkov iz Sobel algoritma k video izhodu v vmesnem pomnilniku,
  - namestitev MDMA za video tok podatkov k Sobel vhodu v vmesnem pomnilniku,
  - namestitev MDMA za tok podatkov iz Sobela k video izhodu v vmesnem pomnilniku.

- **Korak 7: Submit_Video_In_Buffers();** pripravi in odda vhodni video za video dekodirnik ADV7183).
  - Če je definiran ADSPBF561
    - Konfiguriramo začetno lokacijo 2D videa v vmesnem pomnilniku 1 in 2 (za BF561 je 32 bitni prenos, za BF535 pa 16 bitni prenos)
    - Ko je konfiguracija končana, oddamo vhodni video za sistem. **ezErrorCheck(adi_dev_Read(ADV7183DeviceHandle, ADI_DEV_2D,(ADI_DEV_BUFFER*)&Video_In_Buffer2D[i]));**

- **Korak 8: InstallVideoEncoder();** v tem koraku namestimo video kodirnik (ADV 717x) (**ezEnableVideoEncoder()**). Za ADSPBF561 ali ADSPBF535, odvisno kateri je definiran, nastavimo gonilnike za video izhod.
  - konfiguriramo gonilnik video kodirnika,

- odpremo PPI napravo za video izhod,
- posodobimo ADV717x način video delovanja **UpdateADV717xVideoMode (),**
- nastavimo postopek za pretok podatkov,
- omogočimo streaming,
- pripravimo in oddamo video izhod vmesnega pomnilnika **SubmitVideoOutBuffers();**

- **Korak 9: UpdateADV717xVideoMode();** posodobimo ADV717x način video delovanja. Konfiguriramo ADV717x v PAL ali NTSC načinu.

- **Korak 10: SubmitVideoOutBuffers**(); pripravi in odda video izhod vmesnega pomnilnika (za ADV 717x).
  - Če je Sobel omogočen, se odda original video izhod iz vmesnega pomnilnika. Video bo prikazan.
    - Če je definiran BF561,
      - Preveri pavzo
    - Če pavza ni omogočena, se nadaljuje dohodni tok video podatkov. Če je zastavica za pavzo nastavljena (1),
  - Če je definiran BF535 in če je Sobel onemogočen, se prikaže na zaslonu orginalen video okvir. Če je sobel omogočen se odda zajeti okvir video vhoda k Sobel video izhodu. Konfiguracija videa iz vmesnega pomnilnika.

- **Korak 11: EnableVideo();** Omogoči oziroma onemogoči pretok podatkov vhodnega videa (ADV7183).

- **Korak 12: EnableVideoOut(),** Omogoči oziroma onemogoči pretok podatkov vhodnega videa (ADV717x).

- **Korak 13: VideoInCallback();** kliče funkcijo za video vhod (ADV7183).

- **Korak 14: VideoIn**(); Video okvir je bil zajet.

- Če je definiran ADSPBF561:
    - Preveri, če je UpdateBufs_VideoIn zastavica postavljena:
        - Če je, posodobi video vhod/izhod buffer id's **UpdateBufferIDs()**.
    - Preveri, če je video izhod Sobela omogočen in če je pripravljen za zajemanje naslednjega video okvirja:
        - Če je tako, potem je naslov videa v vmesnem pomnilniku ustvarjen za povratni klic kot pArg.
          SobelInInit ((u32 *)pArg);

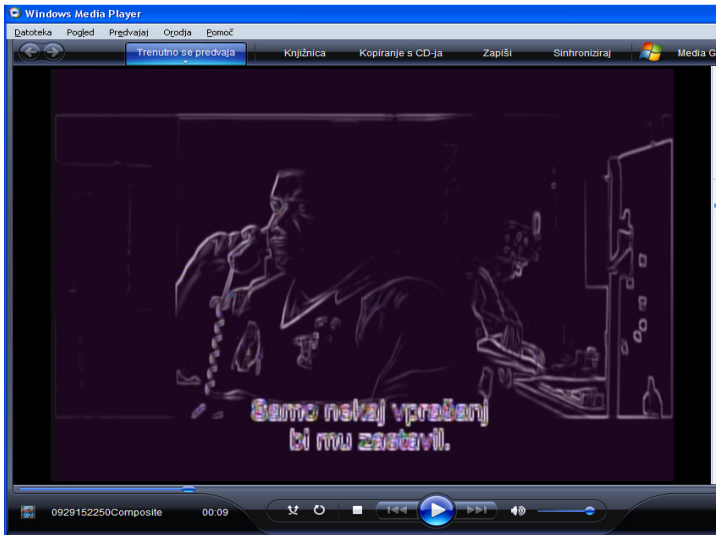
- Če je definiran ADSPBF535:
    - Video okvir je bil zajet, onemogoči se pretok podatkov za video vhod **EnableVideoIn (FALSE).**
    - Zapri ADV7183 **ezErrorCheck(adi_dev_Close(ADV7183DeviceHandle));**
- Ostale funkcije:
    - dohodni video tok je NTSC:
      adi_video_CopyField((char *)(((ADI_DEV_BUFFER *)pPausedBuffer)->TwoD.Data), NTSC_IL,2, TRUE);
    - dohodni video tok je PAL:
      adi_video_CopyField((char *)(((ADI_DEV_BUFFER *)pPausedBuffer)->TwoD.Data), PAL_IL, 2, TRUE);
    - inicializacija sobel algoritma za zaznavanja robov za nov okvir:
      SobelInInit ((u32 *) pPausedBuffer);
    - omogočimo video kodirnik (ADV7171) v BF533Ezkit: **ezEnableVideoEncoder ();**
    - inicializiramo video kodirnik (video izhod - ADV717x):
      InstallVideoEncoder ();
    - omogočimo pretočni video izhod:
      EnableVideoOut (TRUE);
    - vklop vseh led diod:

ezTurnOnAllLEDs();

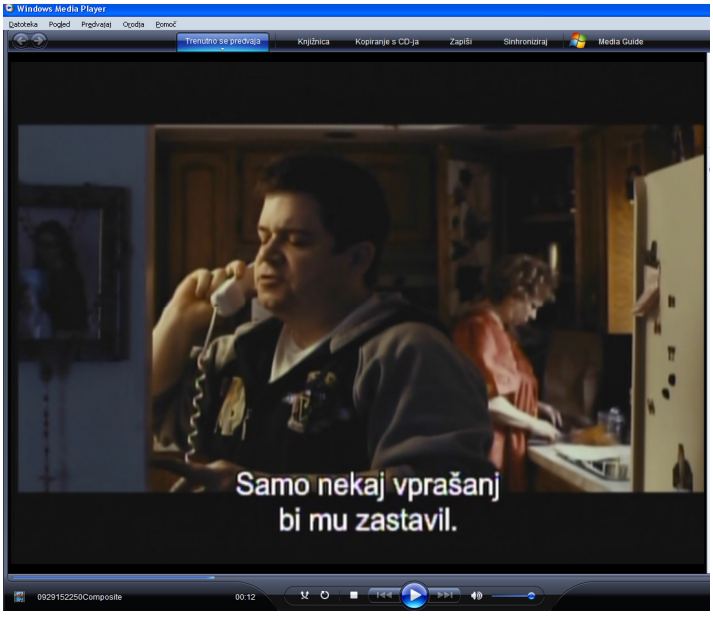
- **Korak 15: Sobel_In_Init();** inicializira vhodni video tok podatkov za vhod sobel algoritma:
    - Video_In_Buf0 je pripravljen za Sobel pretvorbo,
    - Če definiramo BF561, je Video_In_Buf1 pripravljen za Sobel pretvorbo.
    - Inicializacija Sobel vmesnega pomnilnika za upravljanje zastavic in števcev

7 REZULTAT

Kot rezultat diplomskega dela, lahko vidimo zaznavanje robov objektov in besedila v video signalu.



Slika 50: Procesirani video signal



Slika 51: Prvotni video signal

8 SKLEP

Diplomsko nalogo smo zasnovali na obstoječih tehnologijah. Predstavili smo korake od začetka zaznavanja besedila v video signalu do končnega koraka, optičnega prepoznavanja znakov besedila (OCR). Video signal smo pripeljali iz DVD predvajalnika preko Blackfin DSP razvojne plošče na računalnik. Za zaznavanje robov smo uporabili Sobel algoritem, ki išče robove besedila na podlagi odvoda. Izvedli smo tudi podrobno analizo programske kode, ki izvaja procesiranje slike in zaznava robove objektov in besedila v video signalu.

Na samem začetku smo imeli problem najti rešitev, kako poiskati korake procesiranja besedila v digitalnem video signalu, ker je zelo veliko predstavljenih postopkov in smo morali poiskati najbolj učinkovitega.

9 VIRI, LITERATURA

[1] http://archiv.ub.uni-marburg.de/diss/z2007/0107/pdf/djg.pdf

[2] http://dspace.nitrkl.ac.in:8080/dspace/bitstream/2080/1236/1/grover.pdf

[3] http://en.wikipedia.org/wiki/Sobel_operator

[4] http://www.pages.drexel.edu/~weg22/edge.html

[5] http://www.cek.ef.uni-lj.si/magister/nikolic373.pdf

[6] http://eprints.fri.uni-lj.si/1062/1/Diplomsko_delo_-_%C4%8Cerin_A.pdf

10 PRILOGE

10.1 Naslov študenta

Ime in priimek: Marko Kočevar,
Naslov: Krivica 1
Pošta: 3262 Prevorje
E – mail: kocevar.marko@gmail.com

10.2 Življenjepis

Ime in priimek: Marko Kočevar

Rojen: Celje, 27.06.1984

Šolanje: 2008 – 2010
Fakulteta za elektrotehniko, računalništvo in informatiko, smetanova ulica 17, 2000Maribor
VII. stopnja

2003 – 2008
Fakulteta za elektrotehniko, računalništvo in informatiko, smetanova ulica 17, 2000Maribor
VI/II. Stopnja

1999 – 2003
Šolski Center Celje
Pot na Lavo 22, 3000 Celje
V. stopnja

1990 – 1998
Osnovna šola Lesično
Lesično 5 B, 3261 Lesično
II. stopnja

IZJAVA O ISTOVETNOSTI TISKANE IN ELEKTRONSKE VERZIJE DIPLOMSKEGA DELA IN OBJAVI OSEBNIH PODATKOV DIPLOMANTOV

Ime in priimek diplomanta-tke:	Marko Kočevar
Vpisna številka:	93643967
Študijski program:	UNI elektrotehnika – elektronika
Naslov diplomskega dela:	PREPOZNAVANJE BESEDILA V DIGITALNEM VIDEO SIGNALU
Mentor:	izr. prof. dr. Matej Rojc
Somentor:	red. prof. dr. Zdravko Kačič

Podpisani-a Marko Kočevar _izjavljam, da sem za potrebe arhiviranja oddal elektronsko verzijo zaključnega dela v Digitalno knjižnico Univerze v Mariboru.

Diplomsko delo sem izdelal-a sam-a ob pomoči mentorja. V skladu s 1. odstavkom 21. člena Zakona o avtorskih in sorodnih pravicah (Ur. l. RS, št. 16/2007) dovoljujem, da se zgoraj navedeno zaključno delo objavi na portalu Digitalne knjižnice Univerze v Mariboru.

Tiskana verzija diplomskega dela je istovetna elektronski verziji, ki sem jo oddal za objavo v Digitalno knjižnico Univerze v Mariboru.

Podpisani izjavljam, da dovoljujem objavo osebnih podatkov vezanih na zaključek študija (ime, priimek, leto in kraj rojstva, datum diplomiranja, naslov diplomskega dela) na spletnih straneh in v publikacijah UM.

Datum in kraj: 30. 9. 2010

Podpis diplomanta-tke: Kočevar

IZJAVA O USTREZNOSTI DIPLOMSKEGA DELA

Podpisani mentor izr. prof. dr. Matej Rojc izjavljam, da je
(ime in priimek mentorja)
študent Marko Kočevar izdelal diplomsko
(ime in priimek študenta-tke)

delo z naslovom: PREPOZNAVANJE BESEDILA V DIGITALNEM
VIDEO SIGNALU
(naslov diplomskega dela)

v skladu z odobreno temo diplomskega dela, Navodili o pripravi diplomskega dela in mojimi navodili.

Datum in kraj: 30. 9. 2010

Podpis mentorja: