Sveučilište u Zagrebu
Fakultet strojarstva i brodogradnje

ZAVRŠNI RAD

Voditelj rada:
Prof. dr. sc. Bojan Jerbić

Tomislav Šencaj

Zagreb, 2010.

Sažetak

Ovaj završni rad bavi se izradom programa za otkrivanje konture predmeta primjenom metode značajnih razlika. Program mora biti u stanju pomoću web kamere snimati predmete te označiti piksele koji predstavljaju konturu predmeta. Program je pisan u programskom jeziku C koristeći biblioteku OpenCV. Pri izradi programa nisu se koristile gotove funkcije za nalaženje konture objekta (cvCanny, cvFindContours)nego se razvio vlastiti algoritam. Program je zatim proširen kako bi mogao raspoznavati objekte (eng. object recognition). Dobivene konture predmeta nisu se mogle koristiti za raspoznavanje jer se radilo o malom broju piksela zbog toga je korištena segmentacija. Model pozadine dobiven je metodom srednje vrijednosti pozadine, a sama segmentacija je dobivena metodom oduzimanja pozadine. Značajka kojom vršimo prepoznavanje je površina predmeta odnosno broj piksela bijele boje.

Sadržaj

Popis slika i tablica

Izjava

Završni rad izradio sam samostalno, služeći se literaturom i znanjem stečenim tijekom studija. Zahvaljujem se mentoru prof. dr. sc. Bojanu Jerbiću na pomoći i sugestijama prilikom izrade završnog rada. Također se želim zahvaliti asistentu dip. ing Tomislavu Stipančiću na korisnim savjetima i pomoći pri izradi završnog rada. Naposljetku, zahvaljujem svojoj obitelji na podršci, motivaciji i razumijevanju koje su mi pružili tijekom studija.

1. Uvod

1.1 Opis zadatka

Zadatak rada je napisati program koji će pomoću web kamere snimati predmete te označiti piksele koji predstavljaju konturu predmeta. Program mora detektirati predmete pomoću metode značajnih razlika. Metoda se temelji na uspoređivanju vrijednosti susjednih piksela. Pikseli koji zadovoljavaju traženu razliku potrebno je postavi na vrijednost 255 (bijela boja) sve ostale piksele program mora zanemariti, odnosno postaviti na vrijednost 0 (crna boja).

Program je zatim potrebno proširiti kako bi mogao raspoznavati objekte (eng. object recognition).
Pomoću segmentacije treba izolirati predmet od ostatka okoline. Potrebno je stvoriti binarnu sliku na kojoj su pikseli koji predstavljaju pozadinu crne boje (vrijednost nula) , a pikseli koji se odnose na predmet bijele boje(vrijednost 255).

Pomoću dobivene binarne slike vrši se postupak prepoznavanje predmeta. Značajka kojom će se vršiti prepoznavanje je površina predmeta odnosno broj piksela bijele boje. Program uspoređuje površinu predmeta koji se nalazi ispod kamere sa površinama predmeta koji su spremljeni u bazu. Ako nađe jednaku površinu ispisuje poruku o kojem se predmetu radi, inače ispisuje poruku da predmet nije pronađen.

Program se mora napisati u programskom jeziku C/C++ koristeći biblioteku sa vizijskim alatima OpenCV.

1.2 Računalni vid

U današnje vrijeme svjedoci smo brzog razvoja informatičkih tehnologija. Jedno od područja koje se naglo razvija je i računalni vid. Svi mi svakodnevno koristimo blagodati računalnog vida iako nismo svjesni njegove važnosti i rasprostranjenosti. Danas ne možemo pregledavati internet stranice a da ne koristimo usluge računalnog vida. Obična razmjena video sadržaja i slika spada u područje računalnog vida. Računalni vid se koristi i kod izrade računalnih igara. Danas svatko može vidjeti satelitsku snimku svoje kuće, vidjeti kako izgledaju ulice u drugim državama sve zahvaljujući složenim tehnikama računalnog vida kao što su : kalibracija kamere, procesiranje slike, slaganje više slika u jednu cjelinu itd. Zahtjevi mnogih proizvodnih procesa nadilaze sposobnosti ljudskog vida. Proizvodnja se odvija prebrzo ili su tolerancije odstupanja toliko male da ljudsko oko ne može raspoznati loš proizvod. Zbog toga je računalni vid vrlo raširen u modernoj serijskoj proizvodnji za primjerice, pozicioniranje predmeta ili robotske ruke, prepoznavanja uzoraka, upravljanja kakvoćom ... U budućim tehnologijama kao što su: bespilotne letjelice, samoupravljajuća vozila ,medicinska analiza, sigurnosno motrenje računalni vid također će imati važnu ulogu.

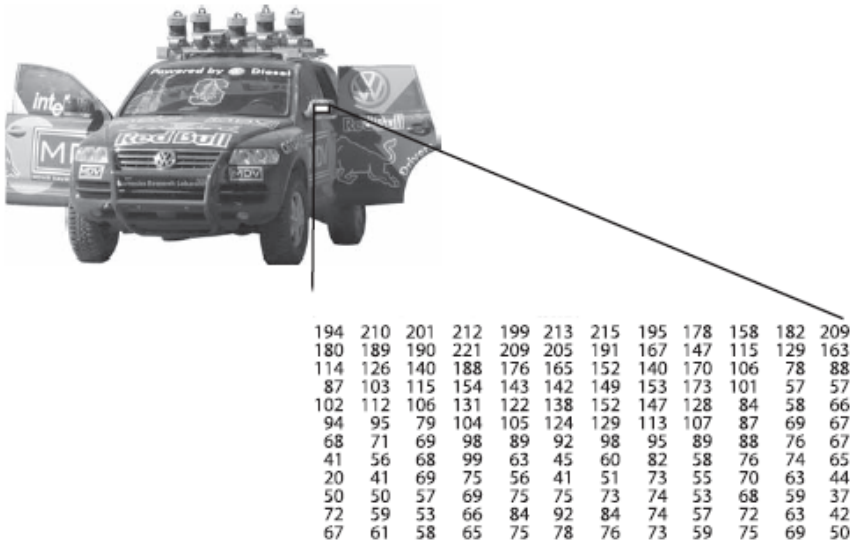
Računalni vid (eng. computer vision) je grana računalne znanosti koja se bavi teorijom i izradom sustava koji služe za prikupljanje informacija iz slike. Slike se mogu prikupljati iz različitih izvora kao što je kamera, određeni medicinski uređaji (CT skener, ultrazvuk), fotografije ... Neke od poddisciplina računalnog vida su praćenje, detekcija i prepoznavanje objekata, detekcija događaja, restauracija slike i slično. Računalni vid je srodan mnogim drugim znanstvenim disciplinama kao što je optika, obrada slike i analiza slike, raspoznavanja uzoraka, robotika, umjetna inteligencija itd.

Biološki i računalni vid su blisko povezani. Biološki vid se bavi istraživanjem vizualnih percepcija ljudi i životinja s ciljem izgradnje modela funkcioniranja tih sustava sa aspekta fizioloških procesa. S druge strane računalni vid se bavi istraživanjem i opisivanjem umjetnih vizijskih sustava koristeći znanja dobivena proučavanjem biološkog vida.

Postoji i pojam „ strojni vid“ (eng. machine vision) koji nije sinonim za računalni vid. Pojam strojni vid definira primjenu računalnog vida u industriji i proizvodnji, zbog toga pojam „strojni vid“ spada u inženjerske discipline. Cilj strojnog vida je da određeni stroj odnosno robot dobije sposobnost „gledanja“. Koristeći strojni vid, strojevi vrše inspekciju proizvoda detektirajući prisutnost, odnosno odsutnost dijelova, uzimajući mjere, čitajući bar kodove... Strojni vid veliku pažnju posvećuje hardveru posebice kamerama.

Kamere koje se najčešće upotrebljavaju u industriji za strojni vid su kompaktne, lagane sa ugrađenom video memorijom i procesorom za obradu slika. Sastoje se od CCD senzora visoke rezolucije sa brzim (DSP) signalnim procesorom za obradu slike. Koristi se RAM memorija za spremanje podataka i slika. Na taj način mnogi problemi računanog vida riješeni su u samoj kameri pomoću hardvera. Osim toga sastoje se od sučelja za komunikaciju sa drugim uređajima koristeći već standardne komunikacijske protokole kao što su Profibus, CAN bus, Industrial Ethernet. Zbog toga se za takve kamere koristi pojam „pametne kamere“ (eng. smart camera) .

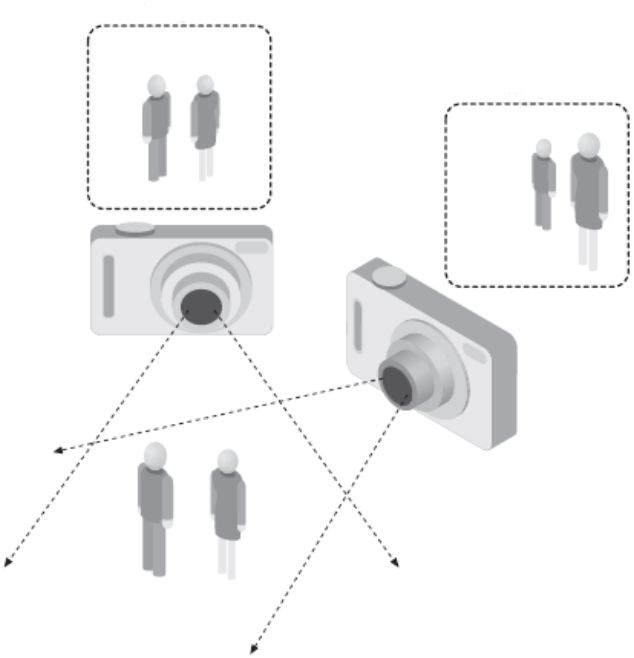
Industrija sustave strojnog vida počinje ozbiljnije primjenjivati tek sredinom 70-tih godina, iako su se prve zamisli upotrebe kamere za nadgledanje proizvodnje pojavile i prije. Sve do početka 80-tih godina razvoj je tekao sporo ali uz doprinos istraživačke zajednice. Sredinom 80 - tih vodeće svjetske automobilske kompanije pokazuju interes za sustave strojnog vida. Krajem 80 – tih i početkom 90 –tih dolazi do porasta interesa zahvaljujući značajnim dostignućima u području digitalizacije slike i pripadajućeg hardvera. Daljnji rast pojavljuje se sredinom 90 – tih pojavom RISC procesora koji su omogućili brzu obradu slike pomoću standardnih računala.



Slika 1-1 Za računalo, auto je tek niz brojeva

U sustavima strojnog vida računalo od kamere prima niz brojeva i ništa više. Slika 1.1 prikazuje sliku automobila. Ljudi na temelju te slike mogu raspoznati određene dijelove automobila, ali računalo „vidi“ samo niz brojeva. Svaki broj osim informacije o slici sadrži i određenu količinu smetnji. Smetnje mogu nastati zbog promjene u osvjetljenju, određenih refleksija, neželjenih pomaka promatranog objekta. Mehanički dijelovi kamere, posebice leća, također uzrokuje određenu distorziju zbog nesavršenosti u izradi. Elektronički dijelovi također uzrokuju određeni električni šum koji utječe na

kvalitetu dobivene slike. Zadatak računalnog vida je izdvojiti korisne informacije iz slike kako bi pomoću određenih algoritama mogli odrediti što slika predstavlja.



Slika 1-2 Slika objekta ovisi o položaju snimanja

Problem dodatno otežava i način na koji kamera prikuplja informacije. Kamera prikuplja informacije iz trodimenzionalnog prostora te ih sprema u obliku brojeva u dvodimenzionalni prostor. Pomoću dobivenih informacija nije moguće u potpunosti rekonstruirati trodimenzionalni sliku. Izgled snimljenog dvodimenzionalnog objekta drastično ovisi o položaju kamere što se može vidjeti na slici 1-2 .

Navedeni problem se može riješiti pomoću kontekstualnih informacija. Što se tiče smetnji njih rješavamo pomoću statističkih metoda. Neke smetnje možemo u potpunosti kompenzirati. Primjerice, distorziju koju uzrokuje leća na kameri je dobro poznata te ju je moguće matematički opisati na temelju mjernih podataka.

2. OpenCV

OpenCV je „ open source“ biblioteka za računalni vid. Biblioteka je pisana u programskom jeziku C i C++ -u te se može koristiti na operativnom sustavu Windows, Linux i Mac OS X. Biblioteku je također moguće koristiti sa programskim jezicima Python, Ruby i Matlab.

Kod dizajniranja OpenCV – a težilo se postići što veću računalnu učinkovitost, pogotovo kod aplikacija u stvarnom vremenu „ real – time application“. Zbog toga je pisan u optimiziranom C – u te je u mogućnosti koristiti prednosti višejezgrenih procesora.
Ako je potrebna dodatna optimizacija može se koristiti Intelova biblioteka IPP (Integrated Performance Primitives). Ova biblioteka sadrži rutine napisane na vrlo niskoj razini što omogućava izradu još bržih i još učinkovitijih aplikacija.

Jedan od ciljeva OpenCV – a je stvaranje jednostavne infrastrukture koja omogućava ljudima brzu izradu složenih vizijskih aplikacija. OpenCV biblioteka sadrži više od 500 funkcija koje pokrivaju područja kao što su: procesiranje slike, kalibracija kamere, stereo vid, vizijski sustavi u robotici, kontrola tolerancija proizvoda... OpenCV također sadrži podbiblioteku za strojno učenje (Mashine Learning Library – MLL).

Projekt OpenCV započeo je u Intelu gdje su razvijali aplikacije u stvarnom vremenu koje traže veliku procesorsku moć. U projekt su se uključila i vodeća sveučilišta kao što je M.I.T gdje su studenti razvijali algoritme namijenjene računalnom vidu. Stručnjaci za optimizaciju računalnog koda također su pridonijeli razvoju OpenCV – a.

Bilo je nekoliko ciljeva koji su se htjeli postići OpenCV – om :

- Razviti ne samo besplatan nego i optimiziran računalni kod za osnovne operacije u računalnom vidu, tako da ih novi korisnik može odmah početi koristiti u svom radu, umjesto da ponovno sam sve piše ispočetka.

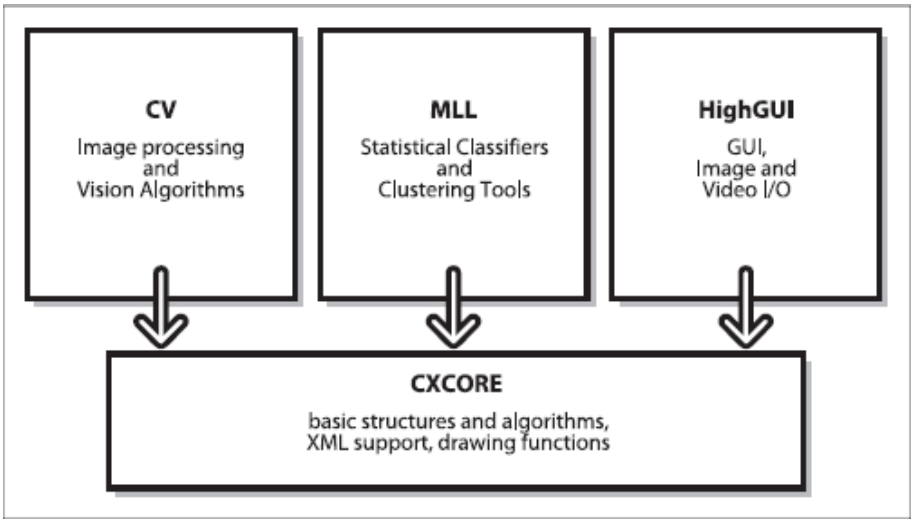
- Pružiti znanje o računalnom vidu osiguravajući jednostavno razvojno okruženje kako bi računalni kod bio lakše čitljiv i prenosiv.

- Razviti napredne aplikacije koje su besplatne ali po kvaliteti i mogućnostima ne zaostaju za komercijalnim rješenjima.

Pojavom višejezgrenih procesora vrijednost OpenCV je još više narasla. Danas mnoge institucije aktivno razvijaju OpenCV za tehnike računalnog vida koje tek dolaze u budućnosti kao što su: percepcija dubine, složeno prepoznavanje uzoraka, integracija kamere sa laserom za procjenu udaljenosti, kalibracija sustava sa više kamera, robotski vid ...

2.1 Struktura i sadržaj OpenCV – a

OpenCV se sastoji od pet glavnih komponenata, četri su prikazane na donjoj slici. CV komponenta sadrži osnove procesiranja slike te vizijske algoritme visoke razine. MLL je komponenta za strojno učenje koja sadrži statističke klasifikatore i alate za raspoređivanje. HighGUI sadrži funkcije i rutine za spremanje i učitavanje slika i videa. Komponenta CXCORE sadrži osnovne podatkovne strukture , funkcije za crtanje grafike , podržavanje XML - a.



Slika 2-1 Struktura OpenCv –a

Na slici se ne nalazi komponenta CvAux koja sadrži eksperimentalne algoritme (pozadinska segmentacija) te prepoznavanje lica pomoću HMM –a (Hidden Markov models).

CvAux pokriva:

- Stereo vid
- Prepoznavanje gesta
- 3 D praćenje
- Kalibracija kamere
- Video nadzor
- Praćenje očiju i usana
- Segmentacija

Neki od ovih alata u budućnosti će postati standardni dijelovi CV komponente.

2.2 Primitivni tipovi podataka u OpenCV – u

Svaki program sadrži u sebi podatke koje obrađuje. Svaki podatak ima dodijeljenu oznaku tipa koje govore o tome kako se dotični podatak pohranjuje u memoriji računala, koji su njegovi dozvoljeni rasponi vrijednosti, kakve su operacije moguće s tim podacima.

OpenCV ima nekoliko tipova primitivnih podataka. Ti podaci nisu primitivni kao u programskom jeziku C, ali mogu se promatrati kao osnovni tipovi podataka u OpenCV – u. Tipovi podataka u OpenCV – u definirani su pomoću struktura.

Strukture su kolekcija od jedne ili više varijabli grupiranih pod jednim imenom da bi se njima lakše manipuliralo. Varijable u strukturama mogu biti različitog tipa i mogu sadržavati bilo koji C tip podataka, uključujući polja i druge strukture.

Najjednostavnija struktura je CvPoint. Struktura CvPoint sadrži dva cjelobrojna člana x i y. U ovu strukturu se spremaju koordinate točke.

Sljedeća struktura koja će se koristiti u programu je CvSize. Sastoji se od dva člana width i height koja su cjelobrojnog tipa. Ako int nije dovoljan može se koristiti struktura CvSize2D32f koja sadrži jednake članove kao struktura CvSize ali float tipa.

Struktura CvSize je kombinacija strukture CvPoint i CvSize te sadrži četiri člana: x, y, width i height.

Svi ovi tipovi podataka imaju svoj konstruktor koji općenito ima isti naziv kao i struktura samo što počinje sa malim početnim slovom. Mora se napomenuti da su ovi „konstruktori“ zapravo funkcije jer se radi o programskom jeziku C ne o C++ -u. Funkcije uzimaju sve potrebne argumente te rezultat spremaju u odgovarajuću strukturu.

Spomenute funkcije su vrlo korisne jer programiranje čine jednostavnim, a program lako čitljivim.
Recimo da želimo nacrtati bijeli pravokutnik sa koordinatama (5, 10) i (20, 30); kod bi izgledao ovako:

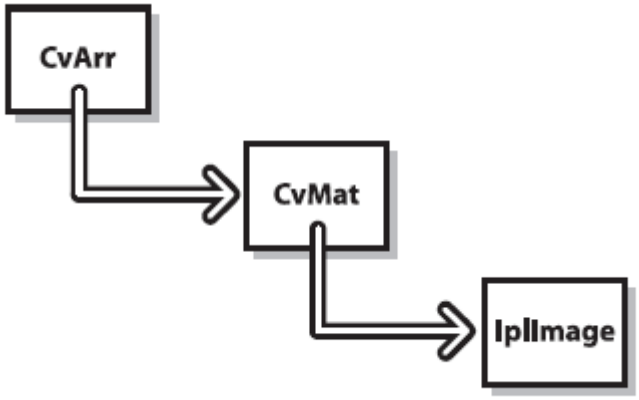
```
cvRectangle(
  mojaSlika,
  cvPoint(5,10),
  cvPoint(20,30),
  cvScalar(255,255,255)
       );
```

Struktura	Sadržaj	Opis
CvPoint	int x, y	Točka u slici
CvPoint2D32f	float x, y	Točka u 2D prostoru
CvPoint3D32f	float x, y, z	Točka u 3D prostoru
CvSize	int width, height	Veličina slike
CvRect	int x, y, width, height	Dio slike
CvScalar	double val[4]	RGBA vrijednosti

Tabela 2-1 Osnovne strukture u OpenCV –u

2.2.1 Matrični tipovi podataka

OpenCV sadrži složenije tipove podataka koje služe za spremanje matrice ili slike. Hijerarhija spomenutih struktura može se vidjeti na slici. Cijeli OpenCV napisan je u programskom jeziku C, ali između struktura postoje odnosi slični nasljeđivanju u programskom jeziku C++. Struktura IplImage izvedena je iz strukture CvMat dok je strukture CvMat izvedena iz strukture CvArr.



Slika 2-1 Objektni dizajn struktura u OpenCv – u

Za rad sa matricama potrebna nam je struktura CvMat. Koncept matrice u OpenCV – u je donekle apstraktniji nego koncept matrice u linearnoj algebri. OpenCv nema poseban način definiranja vektora pa se oni definiraju pomoću matrica dimenzija 3x1 ili 1x3.

Funkcija koja stvara dvodimenzionalnu matricu ima sljedeći prototip:

```
cvMat* cvCreateMat ( int rows, int cols, int type );
```

Podacima u matrici možemo pristupati direktno preko strukture CvMat ili pomoću određenih funkcija . Primjerice ako želimo saznati veličinu matrice možemo koristiti funkciju cvGetSize(CvMat*) koja vraća CvSize strukturu.

2.3 IplImage struktura

IplImage je osnovna struktura za rad sa slikama. Slika može biti crno – bijela, u boji, četri kanalna (RGB + alfa). OpenCV sadrži mnoge operacije koje možemo koristiti na slikama. Primjerice, alate za mijenjanje veličine slike, izvlačenje pojedinih kanala, traženje najmanje ili najveće veličine pojedinog kanala, zbrajanje dvije slike itd. Strogo gledano riječ je zapravo o CvMat strukturi ali sa određenim dodacima kojima se slika interpretira kao matrica. Struktura je originalno definirana kao dio Intelove biblioteke Image Processing Library (IPL). Točna definicija IplImage strukture je sljedeća:

```
typedef struct _IplImage {
  int                  nSize;
  int                  ID;
  int                  nChannels;
  int                  alphaChannel;
  int                  depth;
  char                 colorModel[4];
  char                 channelSeq[4];
  int                  dataOrder;
  int                  origin;
  int                  align;
  int                  width;
  int                  height;
  struct _IplROI*      roi;
  struct _IplImage*    maskROI;
  void*                imageId;
  struct _IplTileInfo* tileInfo;
  int                  imageSize;
  char*                imageData;
  int                  widthStep;
  int                  BorderMode[4];
  int                  BorderConst[4];
  char*                imageDataOrigin;
} IplImage;
```

Varijable width i height sadrže informacije o rezoluciji slike. Varijabla nChannels može imati vrijednosti 1, 2, 3 ili 4. Ako je slika crno – bijela tada ima jedan kanal te će varijabla nChannels imati vrijednost 1. Slika u boji ima tri kanala. Postoji i četvrti kanal koji je određen za prozirnost slike takozvanu alphu.

Varijabla depth može imati šest različitih vrijednosti koje su definirane u zaglavnoj datoteci ipl.h. Vrijednosti definiraju rezoluciju piksela na slici. Na primjer, ako je vrijednost varijable depth = IPL_DEPTH_8U nijanse boje za pojedini piksel se mogu definirati brojčanom vrijednošću od 0 do 255.
Moguće vrijednosti varijable depth prikazane su u tablici

Macro	Rezolucija slike
IPL_DEPTH_8U	Unsigned 8-bit integer (8u)
IPL_DEPTH_8S	Signed 8-bit integer (8s)
IPL_DEPTH_16S	Signed 16-bit integer (16s)
IPL_DEPTH_32S	Signed 32-bit integer (32s)
IPL_DEPTH_32F	32-bit floating-point single-precision (32f)
IPL_DEPTH_64F	64-bit fl oating-point double-precision (64f)

Tabela 2-2 Tipovi slike

Sljedeća važna varijabla je dataOrder. Ona može imati dvije vrijednosti IPL_DATA_ORDER_PIXEL ili IPL_DATA_ORDER_PLANE. Ako želimo biti precizniji jedino je prva vrijednost podržana od strane OpenCV – a, ali obje vrijednosti su podržane od strane IPL/IPP –a (Image Processing Librarry / Integrated Performance Primitives). Vrijednost IPL_DATA_ORDER_PIXEL definira način spremanja vrijednosti boje za pojedini piksel u podatkovno polje imageData. Svaki piksel je definiran sa tri vrijednosti ako se radi o slici u boji (RGB). Na prva tri mjesta dolaze podaci o boji za prvi piksel, zatim za drugi i tako dalje sve do kraja prvog reda, nakon toga postupak se ponavlja za sve ostale redke.

Vrijednost IPL_DATA_ORDER_PLANE podatke organizira u ravnine, svaki kanal predstavlja jednu ravninu koje se slažu jedna na drugu.

widthStep

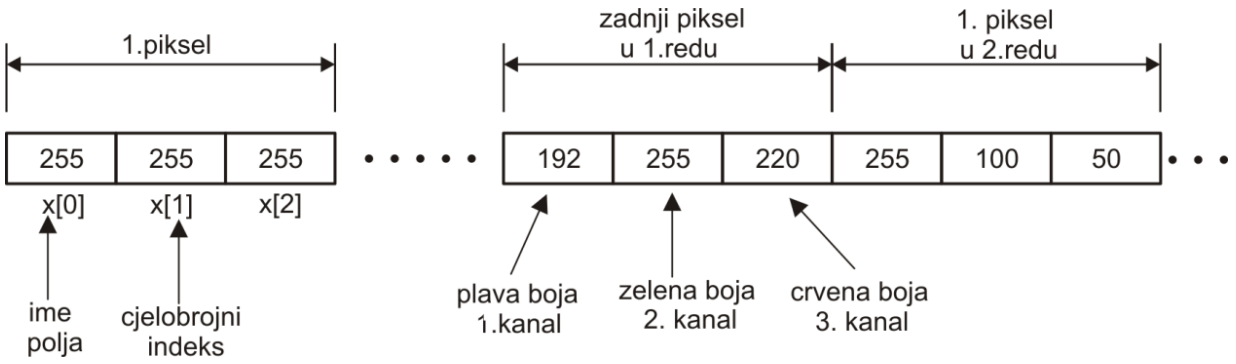
Već je bilo spomenuto da se u polje upisuju prvo vrijednosti piksela iz prvog reda slike zatim dolazi drugi red i tako dalje. Za spremanje vrijednosti potrebna je određena količina memorije. Broj bajtova potrebnih za upis vrijednosti piksela iz prvog reda spremljen je u varijablu widthStep. Ova varijabla je vrlo korisna kada je potrebno pristupiti pojedinom pikselu na slici.

imageData

Slika sadrži veliku količinu podataka pa bi bilo nespretno za svaki pojedini piksel koristiti zasebnu varijablu. Zbog toga se podaci smještaju pod isto nazivlje, a pojedine podatke dohvaćamo pomoću brojčanog indeksa. Ovakav način obrade podataka omogućavaju polja podataka (kraće polja , eng. arrays).

Polje je niz konačnog broja istovrsnih podataka . Ti podaci mogu biti bilo kojeg tipa, ugrađenog (primjerice int ili float) ili korisnički definiranog. Pojedini članovi polja mogu se dohvatiti pomoću cjelobrojnog indeksa te mijenjati neovisno o ostalim članovima polja. OpenCV sliku sprema u jednodimenzionalno polje.

Kod jednodimenzionalnog polja članovi su složeni u linearnom slijedu, a indeks pojedinog člana odgovara njegovoj udaljenosti od početnog člana. Nakon što deklariramo polje prevoditelj će osigurati određenu količinu memorije potrebnu za spremanje polja. Deklaracijom članovi polja nisu inicijalizirani tako da imaju slučajne vrijednosti. Na slici može se vidjeti primjer polja za spremanje slike uz uvjet da varijabla dataOrder ima vrijednost IPL_DATA_ORDER_PIXEL.



Slika 2-2 Način spremanja slike u polje

Polju pristupamo pomoću varijable dataImage koja je zapravo pokazivač na spomenuto polje. Adresa bilo koje varijable je zapravo broj i s njom možemo postupati kao i sa svakim drugim brojem. Ako znate adresu varijable možete kreirati drugu varijablu u koju ćete staviti adresu prve varijable. To se upravo radi sa pokazivačima, oni sadrže adrese drugih varijabli. Različiti tipovi podataka zauzimaju različitu količinu memorije u računalu. Svaki bajt u memoriji ima svoju vlastitu, jedinstvenu adresu, tako da višebajtne varijable zauzimaju nekoliko adresa. Pokazivači kod višebajtnih podatak pokazuju na adresu prvog bajta kojeg oni zauzimaju.

Postoji više načina pristupanja pojedinom pikselu. Jedan od najlakših načina je pomoću funkcije cvGet2D ( ).

```
CvScalar cvGet2D(const CvArr* arr,
                 int row,
                 int col ) ;
```

Funkcija traži tri argumenta. Prvi argument je pokazivač na polje koje sadrži tražene vrijednosti piksela, preostala dva argumenta su koordinate željenog piksela.

Funkcija cvGet2D( ) vraća C strukturu CvScalar definiranu kao:

```
typdef struct CvScalar
{
       double val[4];
}
CvScalar;
```

Vrijednost piksela za pojedini kanal sprema se u polje val [i]. Ako je riječ o crno – bijeloj slici polje val [0] sadrži vrijednost osvjetljenja za pojedini piksel. Ostale tri vrijednosti su postavljene na nulu. Ako se radi o slici u boji tada prvi član polja val [0] sadrži vrijednost piksela za plavu boju, u drugi član spremljena je vrijednost zelene boje a u treći vrijednost crvene boje.

Vrijednost piksela možemo mijenjati pomoću funkcije cvSet2D ( ). Prototip funkcije izgleda ovako:

```
void cvSet2D(CvArr* arr,
                 int row,
                 int col,
                 CvScalar value);
```

Funkcije cvGet2D ( ) i cvSet2D( ) su jednostavne za korištenje ali pomoću njih možemo pristupiti samo jednom pikselu. Ako želimo pristupiti većem broju piksela tada moramo čitati vrijednosti direktno iz polja u kojem su spremljene tražene vrijednosti. S obzirom da ćemo raditi sa slikama,vrijednosti piksela su spremljene u polju imageData koje se nalazi u strukturi IplImage.

```
step =frame->widthStep;
channels = frame->nChannels;
data = (uchar *)frame->imageData;

for(i=0;i< (height-1);i++)

      {

          for(j=0;j<(width-1);j++)
              {
          plava = data[i*step+j*channels];
          zelena = data[i*step+j*channels+1];
          crvena = data[i*step+j*channels+2];
               }
}
```

Gore je prikazan dio koda pomoću kojeg pristupamo svim pikselima spremljenima u polje data odnosno imageData.

3. Opis programa za detekciju konture i prepoznavanje predmeta

Potrebno je napisati program za otkrivanje kontura objekta primjenom metode značajnih razlika. Program mora biti u stanju pomoću web kamere snimati predmete te označiti piksele koji predstavljaju konturu predmeta. Za provjeru rada programa korištena je kamera Logitech Webcam 200. Program je pisan u programskom jeziku C koristeći biblioteku sa vizijskim alatima OpenCV. Kao razvojno okruženje korišten je program Microsoft Visual C++ 2008 Express Editon.

Pri izradi programa nisu se koristile gotove funkcije za nalaženje konture objekta (cvCany, cvFindContours)nego se razvio vlastiti algoritam. Algoritam uspoređuje vrijednosti susjednih piksela te ako je razlika dovoljno velika piksele označava bijelom bojom. Pikseli koji ne zadovoljavaju traženi uvjet alogritam zanemaruje, odnosno postavlja vrijednost na nulu što odgovara crnoj boji. Kao rezultat dobije se video kojim možemo otkriti konturu željenog predmeta. Obzirom da je pisan vlastiti algoritam program je osjetljiv na promjene osvjetljenja i smetnje koje nastaju zbog korištenja jeftine kamere. Također se može dogoditi da program detektira sjenu kao konturu predmeta.

Program je zatim proširen kako bi mogao raspoznavati objekte (eng. object recognition). Dobivene konture predmeta nisu se mogle koristiti za raspoznavanje jer se radilo o malom broju piksela. Zbog toga je korištena segmentacija. Pomoću metoda segmentacije možemo izolirati predmet od ostatka okoline. Korištena je metoda srednje vrijednosti pozadine. Nakon što program stvori model pozadine u stanju je prepoznati željeni predmet na slici. Program stvara binarnu sliku na kojoj su pikseli koji predstavljaju pozadinu crne boje (vrijednost nula) , a pikseli koji se odnose na predmet dobivaju vrijednost 255 (bijela boja).

Pomoću dobivene binarne slike vrši se postupak prepoznavanje predmeta. Značajka kojom vršimo prepoznavanje je površina predmeta odnosno broj piksela bijele boje. Riječ je od jednostavnoj metodi koja je pokazala solidne rezultate. Najveća joj je mana što nije u mogućnosti prepoznati predmete jednake površine. Najprije moramo spremiti predmete koje želimo raspoznavati. Program omogućava spremanje do četri predmeta. Nakon toga može se raditi postupak prepoznavanja. Program uspoređuje površinu predmeta koji se nalazi ispod kamere sa površinama predmeta koji su spremljeni u bazu. Ako nađe jednaku površinu ispisuje poruku o kojem se predmetu radi, inače ispisuje poruku da predmet nije pronađen.

3.1 Detekcija konture

```
#include<cv.h>
#include<highgui.h>
#include <stdio.h>
```

Naputak #include govori C prevoditelju da u programski kod doda sadržaj određene datoteke tijekom prevođenja programa. U tim datotekama nalaze se određene funkcije. Postoje dvije vrste funkcija: funkcije iz funkcijske biblioteke (function librarys), koje su sastavni dio prevoditelja ili OpenCV – a i funkcije koje definira programer (use – defined functions). Naredbom #include zapravo kažemo prevoditelju gdje mora tražiti funkcije koje smo koristili u programu.

Ako je ime datoteke napisano ovako: < zaglavna_datoteka.h >, pretprocesor najprije traži datoteku u standardnom include direktoriju. Ako datoteka tamo nije nađena, pretprocesor traži datoteku u direktoriju gdje se nalazi i izvorni kod. Postoji i drugi način a to je da ime datoteke stavimo pod navodne znakove : #include " moja_datoteka.h" . U tom slučaju pretprocesor ne pretražuje standardni direktorij nego traži datoteku u direktoriju gdje se nalazi izvorni kod.

Za potrebe programa prvo je potrebna zaglavna datoteka stdio.h. U njoj se nalaze osnovne funkcije za ispis informacija na ekranu kao što je funkcija printf() . Također sadrži i funkciju scanf() koja očitava podatke koje korisnik unese tipkovnicom i pridružuje ih jednoj od varijabli programa.

OpenCV također ima svoje zaglavne datoteke. Za rad programa trebt će nam zaglavna datoteka cv.h i highgui.h. U zaglavnoj datoteci cv.h nalaze se funkcije kao što su cvSmooth, cvThreshold ...
U zaglavnoj datoteci highgui.h nalaze se funkcije za učitavanje slika i videa, primjerice funkcije cvLoadCameraCapture(), cvLoadImage(), te funkcije za izradu grafičkih elemenata kao što su prozori cvNamedWindow() i slične funkcije.

```
int main( int argc, char** argv ) {
//...
     }
```

Oznaka main() je naziv za glavnu funkciju (eng. Main – glavni). Svaki programski jezik pisan u C – u mora imati točno jednu main() funkciju. Izvođenje svakog programa počinje naredbama koje se nalaze u njoj. Riječ je o simboličkom imenu koje daje do znanja prevoditelju koji se dio programa treba prvo početi izvoditi. Glavna funkcija izvršava određene naredbe i funkcije te daje odgovarajući rezultat. Podatkovni tip rezultata definira se ispred glavne funkcije oznakom int . Int dolazi od engleske riječi interger što znači cijeli broj. Drugim riječima int ukazuje da će rezultat glavne funkcije biti cijeli broj. Budući da se glavni program pokreće iz operacijskog sustava (DOS, UNIX, Windows), rezultat glavnog programa se vraća operacijskom sustavu. Najčešće je kod koji signalizira pogrešku nastalu tijekom izvođenja programa ili obavijest o uspješnom izvođenju.

Iza riječi main dolazi par otvorena - zatvorena zagrada. U njoj se nalaze podaci koje operacijski sustav prenosi programu. Ti podaci nazivaju se argumenti ili parametri funkcije. Ako ne unosimo nikakve podatke programu tada je dovoljno zagrade ostaviti praznim.

Prvi argument argc je broj parametara koji se prenose programu iz radne okoline. Parametrima možemo pristupiti pomoću članova polja argv, od argv[0] do argv [argc – 1]. Ti članovi su pokazivači na znakovne nizove koji sadrže pojedine parametre. Član agrv[0] je pokazivač koji pokazuje na ime kojim je program pozvan. Na kraju imamo par otvorena – zatvorena vitičasta zagrada. Unutar tih zagrada upisujemo naredbe ili druge funkcije.

```
 int height,width,step,channels;

 int plava,zelena,crvena;

 int i,j;
   i=j=0;
 char c;

 int stepr, channelsr;

 int razlika;

uchar *data,*datar;

int plava_plus,zelena_plus,crvena_plus;
int plava_minus,zelena_minus,crvena_minus;
```

Nakon deklaracije main() funkcije moramo deklarirati varijable koje koristimo u programu. Deklaracijom memorijskom prostoru u koji će se pohranjivati vrijednosti varijabli dodjeljujemo simboličko ime koje lako pamtimo. Pri prevođenju programa prevoditelj će zapamtiti njihova imena, te za njih rezervirati odgovarajući prostor u memoriji računala. Varijable postaju realne tek kada im se pokuša pristupiti, primjerice kada im se pridruži određena vrijednost. Tek tada prevoditelj rezervira određeni memorijski prostor u koji sprema željene podatke.

Ispred imena varijable nalazi se oznaka koja definira tip varijable. Većina varijabli u programu su cjelobrojnog tipa oznaka int. Oznaka int kaže prevoditelju da rezervira 2 bajta memorijskog prostora u koji se može pohraniti brojčana vrijednost u rasponu od -32768 do 32767. Ostali tipovi varijable su primjerice long (Long integer), float (Single- precision floating point), char (Charakter). Za potrebe programa koristit će se i tip varijable uchar (Usingned charcter). Riječ je tipu varijable koja zauzima 1 bajt memorije, a u nju se može spremiti brojčana vrijednost u rasponu od 0 do 255.

U varijable height i width spremamo rezoluciju videa. Varijabla step sadrži broj bajtova jednog reda piksela a varijabla channels sadrži broja kanala.

Varijable plava, zelena, crvena služe za spremanje vrijednosti boje za pojedini piksel.

Varijable i, j se moraju inicijalizirati na nulu jer se korist kod for petlji. Varijable stepr i channelsr sadrže informacije vezane za generiranje videa na kojem se vide konture.

Zadnje varijable služe za spremanje graničnih vrijednosti boja pojedinih piksela kojima se pomoću if naredbe određuju pikseli koji predstavljaju konturu predmeta.

Kod rada sa videom koristi se struktura CvCapture. Struktura sadrži informacije potrebne za čitanje frameova sa kamere ili iz video datoteke. Ovisno o izvoru, strukturu CvCapture možemo pozvati i inicijalizirati na dva načina :

```
CvCapture* cvCreateFileCapture( const char* filename );
    CvCapture* cvCreateCameraCapture( int index );
```

Ako želimo učitavati video iz određene datoteke spremljene na disku tada koristimo prvu funkciju. Funkcija traži samo jedan argument i to ime datoteke zajedno sa ekstenzijom zapisano u obliku stringa. Ako se datoteka ne nalazi u istom direktoriju kao i program tada je potrebno upisati cijeli path do željene datoteke.

Ako se datoteka zbog nekog razloga ne može otvoriti (npr. tražena datoteka ne postoji) funkcija vraća pokazivač vrijednosti NULL, ali pokazivač će također imati vrijednost NULL ako nije poznat kodek kojim je kompresiran video. Primjerice, ako želimo čitati datoteku kodiranu sa DIVX ili MPG4 kompresijom potrebno je imati odgovarajući DLL datoteku koja osigurava potrebno dekodiranje. Zbog toga je važno provjeriti vrijednost koju vraća funkcija cvCreateFileCapture(), jer iako funkcija radi na jednom računalu na drugom možda neće raditi jer nedostaje potrebna DLL datoteka.

Za učitavanje videa sa kamere koristi se funkcija cvCreateCameraCapture() koja se bitno ne razlikuje od spomenute funkcije cvCreateFileCapture( ). Funkcija samo traži identifikator koji kojoj kaže kojoj kameri treba pristupiti. Identifikator je običan broj koji iznosi nula ako se radi o samo jednoj kameri te se povećava za određeni inkrement ako se radi o sustavu sa više kamera. Drugi dio identifikatora je takozvana „domena“ (eng. domain) kamere. Njome određujemo o kakvoj se kameri radi. Domena može biti bilo koja konstanta prikazana u tablici 3-1 .

Makro naredba	Numerička vrijednost
CV_CAP_ANY	0
CV_CAP_MIL	100
CV_CAP_VFW	200
CV_CAP_V4L	200
CV_CAP_V4L2	200
CV_CAP_FIREWIRE	300
CV_CAP_IEEE1394	300
CV_CAP_DC1394	300
CV_CAP_CMU1394	300

Tabela 3-1 Makro naredbe za funkciju cvCreateCameraCapture( )

U većini slučajeva definiranje domene nije potrebno kada radimo samo sa jednom kamerom. Postoji mogućnost da se argument funkcije cvCreateFileCapture postavi na vrijednost -1. U tom slučaju otvara se novi prozor u kojem korisnik može odabrati željenu kameru.

```
printf("\n Kolika je razlika izmedu piksela?: ");
     scanf("%d",&diff);
```

Funkcija scanf( ) omogućava da korisnik sam definira željenu razliku između susjednih piksela.

```
CvSize     size     =     cvSize(     (int)cvGetCaptureProperty(     capture,
CV_CAP_PROP_FRAME_WIDTH,
               (int)cvGetCaptureProperty(capture, CV_CAP_PROP_FRAME_HEIGHT));
```

Struktura CvCapture sadrži mnoge informacije o videu koji se nalazi u njoj. Tim informacijama pristupamo pomoću funkcije cvGetCaptureProperty ( ). Prvi argument funkcije je ime varijable koja sadrži strukturu CvCapture. Drugim argumentom određujemo koje svojstvo želimo saznati. Popis mogućih svojstva prikazan je u tablici. Potreba rezolucija se sprema u strukturu CvSize pomoću funkcije cvSize ( ).

Makro naredba	Numerička vrijednost
CV_CAP_PROP_POS_MSEC	0
CV_CAP_PROP_POS_FRAME	1
CV_CAP_PROP_POS_AVI_RATIO	2
CV_CAP_PROP_FRAME_WIDTH	3
CV_CAP_PROP_FRAME_HEIGHT	4
CV_CAP_PROP_FPS	5
CV_CAP_PROP_FOURCC	6
CV_CAP_PROP_FRAME_COUNT	7

Tabela 3-2 Makro naredbe u funkciji cvGetCapturePropertiy( )

Ako nam se određeno svojstvo ne odgovara možemo koristiti funkciju cvSetCaptureProperty() kojom sami možemo definirati željeno svojstvo. Prototip funkcije izgleda ovako:

```
int cvSetCaptureProperty(
  CvCapture* capture,
int          property_id,
double       value
     );
```

Nakon što smo definirali željene parametre videa stvaramo sliku pomoću fukcije cvCreateImage()

```
IplImage* rezultat  = cvCreateImage( size,IPL_DEPTH_8U, 1);
   IplImage* siva   = cvCreateImage( size,IPL_DEPTH_8U, 1);
             Imask = cvCreateImage( size, IPL_DEPTH_8U, 1 );
   IplImage* foreground  = cvCreateImage( size,IPL_DEPTH_8U, 1);
   cvZero(foreground);
        cvZero(Imask);
```

Funkcija cvCreateImage () traži tri argumenta. Na mjesto prvog argumenta dolazi informacija o rezoluciji videa koju smo saznali funkcijom svSetCaptureProperty ( ).Trećim argumentom definiramo broj kanala slike. Ako je treći kanal 1, kao u našem slučaju, radi se o crno - bijeloj slici. Ako želimo stvoriti sliku u boji treći argument mora biti 3 jer svaki kanal predstavlja jednu osnovnu boju (RGB). Postoji i četvrti kanal kojim određujemo prozirnost slike, tada treći argument mora biti četiri.

Funkcijom cvZero() vrijednost matrice odnosno polja postavljamo na nulu. U sliku Imask spremat će se rezultat segmentacije.

```
cvNamedWindow( "Kamera", CV_WINDOW_AUTOSIZE );
cvNamedWindow( "Kontura", CV_WINDOW_AUTOSIZE );
cvNamedWindow( "Segmentacija", CV_WINDOW_AUTOSIZE );
```

OpenCV funkcije koje omogućavaju interakciju sa operacijskim sustavom , hardverom kao što je kamera, spremljene su u biblioteku zvanu HighGUI (high – level graphical user inerface). HighGUI omogućava stvaranje prozora, prikaz slike, čitanje i spremanje videa ili slike, interakciju sa mišem i tipkovnicom. Funkcija kojom prikazujemo sliku na ekranu računala je cvNamedWindow(). Točni prototip funkcije izgleda ovako:

```
int cvNamedWindow(
    const char* name,
    int flags = CV_WINDOW_AUTOSIZE
);
```

Prvim argumentom definiramo ime prozora. Drugi argument funkcije može biti nula ili CV_WINDOW_AUTOSIZE koji je definiran kao osnovni. Ako je drugi argument postavljen na CV_WINDOW_AUTOSIZE tada HighGUI prilagođava prozor slici. Ako se učita nova slika prozor će se automatski prilagoditi njezinoj rezoluciji, ali korisnik ne može dodatno mijenjati veličinu prozora. Ako automatska prilagodba prozora nije potrebna tada vrijednost drugog argumenta može biti nula; tada korisnik može mijenjati veličinu prozora prema želji.

Program će prikazivati tri prozora . Prvi prozor imena Kamera prikazivat će sliku koju dobijemo kamerom. U prozoru Kontura prikazivat će se dobivena kontura predmeta, a u prozoru Segmentacija rezultat segmentacije.

```
frame = cvQueryFrame( capture );
          if( !frame ) break;
```

Nakon što je struktura CvCapture definirana potrebno je iz nje čitati framove. Postoji više metoda ali u programi će se koristiti jednostavnija metoda pomoću funkcije cvQueryFrame ( ). Ova funkcija je kombinacija funkcija cvGrabeFrame ( ) i cvRetrieveFrame ( ) te vraća pokazivač IplImage*. If naredbom provjeravamo je li funkcija cvQureryFrame() uspješno spremila frame, ako je došlo do određene greške izlazimo iz beskonačne petlje naredbom break.

Kako bi mogli uspoređivati piksele moramo bit u mogućnosti pristupiti svakom pojedinom pikselu. Za pristupanje pojedinom pikselu potrebne su nam određene informacije koje se nalaze u strukturi IplImage naziva frame. Rezolucija videa koja se nalazi u varijablama height i width spremamo u nove varijable istog naziva. Vrijednosti pojedinog piksela koje su spremljene u varijabli imageData spremamo u novu varijablu naziva data. Broj kanala spremamo u varijablu channels , a informacija o količini memorije koju zauzimaju pikseli pojedinog reda koja se nalazi u varijabli widthStep spremamo u varijablu step. To sve ponavljamo za sliku „rezultat“ u kojoj će se prikazivati tražena kontura.

```
height = frame->height;
width = frame->width;
step =frame->widthStep;
channels = frame->nChannels;
data = (uchar *)frame->imageData;

stepr=rezultat->widthStep;
channelsr=rezultat->nChannels;
     datar = (uchar *)rezultat->imageData;
```

Kada smo odredili sve potrebne informacije koristimo ugniježđenu for petlju kojom pristupamo pojedinom pikselu radi usporedbe vrijednosti .

```
for(i=0;i< (height-1);i++)

      {
```

```
            for(j=0;j<(width-1);j++)

          plava = data[i*step+j*channels];
          zelena = data[i*step+j*channels+1];
               crvena = data[i*step+j*channels+2];
```

Trenutnu vrijednost piksela spremamo u varijable plava, zelena i crvena, za svaki kanal po jedna varijabla.

```
plava_plus = data[(i+1)*step+(j+1)*channels]+diff;
zelena_plus = data[(i+1)*step+(j+1)*channels+1]+diff;
crvena_plus = data[(i+1)*step+(j+1)*channels+2]+diff;

plava_minus = data[(i+1)*step+(j+1)*channels]-diff;
zelena_minus = data[(i+1)*step+(j+1)*channels+1]-diff;
     crvena_minus = data[(i+1)*step+(j+1)*channels+2]-diff;
```

Potrebno je izračunati granične vrijednoti pojednog piksela kako bi mogli odlučiti koji pikseli predstavljaju konturu predmeta. Gornja granična vrijednost sprema se u varijable plava_plus, zelena_plus, crvena_plus. Donja granična vrijednost za susjedni piksel spremljena je u varijable plava_minus, zelena_minus, crvena_minus.

```
if( (plava > plava_plus || plava < plava_minus )

   &&( zelena > zelena_plus || zelena < zelena_minus )

   &&( crvena > crvena_plus || crvena < crvena_minus ))

   {
        datar[(i)*stepr+(j)*channelsr]=255;
   }

    else datar[(i)*stepr+(j)*channelsr]=0;

        }
         }
```

Zatim dolazi naredba if kojom uspoređujemo vrijednosti susjednih piksela. Prvo za svaki kanal zasebno provjeravamo je li vrijednost trenutnog piksela veća od gornje granične vrijednosti ili manja od donje granične vrijednosti susjednog piksela. Ako je ovaj uvijet zadovoljen za sva tri kanala (logička operacija &) if naredba je zdovoljena te se u polju datar vrijednost piksela postavlja na 255 što odgovara bijeloj boji. Ako IF naredba nije zadovoljena vrijednost piksela postavlja se na nulu što odgovara crnoj boji. Na taj način napravili smo određenu segmentaciju, piksele koji predstavljaju konturu označili smo bijelom bojom.

```
                cvShowImage( "Kamera", frame );
                cvShowImage( "Kontura", rezultat);
                     cvShowImage( "Segmentacija", Imask );
```

Funkcijom cvNamedWindow() stvorili smo prozor ali ova funkcija ne stavlja nikakve podatke unutar prozora samo stvara grafički dio prozora. Da bi učitali podatke u prozor koristimo funkciju cvShowImage(). Prototip funkcije cvShowImage() je:

```
                    void cvShowImage(
```

```
    const char* name,
    const CvArr* image
    );
```

Prvi argument je naziv prozora u koji želimo učitati podatke, na mjesto drugog argumenta dolazi varijabla koja sadrži željene podatke.

```
c = cvWaitKey(100);
if(c == 27 )
    break;
```

Funkcijom cvWaitKey() zaustavljamo izvršenje programa na određeno vrijeme. Vrijeme koje želimo da program čeka definiramo pomoću argumenta, osnovana mjerna jedinica je milisekunda. Ako pritisnemo tipku unutar definiranog vremena funkcija vraća ASCII vrijednost pritisnute tipke, inače vraća nulu.

```
switch(c)
    {
        case 's':

            zapamtiPredmet(Imask);

            cvWaitKey(100);
            break;

        case 'u':
            usporedivanje(Imask);

            break;
        case 'b':

            brisanje();
            break;


        }
```

Određeni dijelovi programa pokreću se pomoću tipkovnice, zbog toga koristimo naredbu switch () koja je idealna za izradu programskih menija. Naredba switch( ) dopušta višestruke izlaze iz funkcije, takozvano grananje, a efikasnija je i lakša za upotrebu nego ugniježđena if naredba. Ova naredba ispituje izraz i izlazi na mjestu koje se poklapa s izlaznim rezultatom. Ako nema takvog rezultata , kontrola se prepušta unaprijed određenoj vrijednosti, a ako ne postoji niti ona, izlazi se iz naredbe switch i program se nastavlja dalje.

Ako korisnik pritisne tipku „ s “ program će izvršiti funkciju zapamtiPredmet () zatim će čekati određeno vrijeme i na kraju izaći iz naredbe switch. Funkcijom zapamtiPredmet () spremamo u memoriju željeni predmet. Tipkom „ u “ uspoređujemo spremljene predmete sa trenutnim predmetom. Ako se pritisne tipka „ b “ poziva se funkcija brisanje() kojom brišemo memoriju u kojoj su spremljeni željeni predmeti.

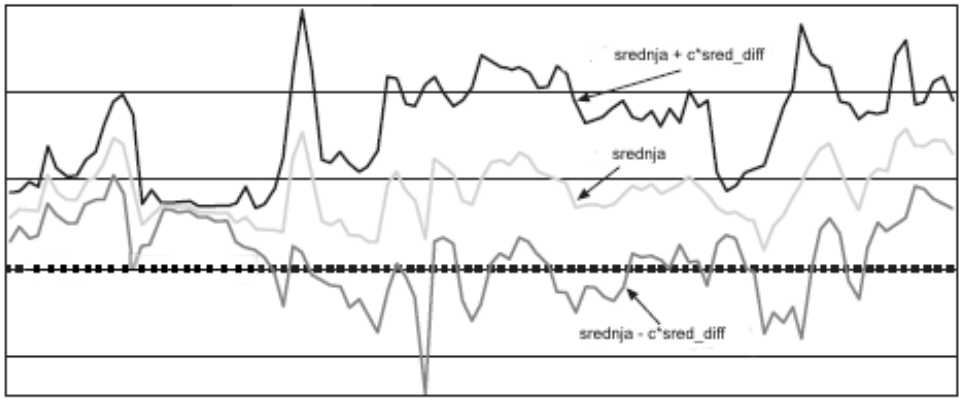
Za spremanje slike, u našem primjeru riječ je zapravo o framu videa koristi se funkcija cvSaveImage(). Prototip funkcije izgleda ovako:

```
int cvSaveImage(
    const char* filename,
    const CvArr* image
    );
```

Prvim argumentom definiramo naziv i format slike. Funkcija podržava sve važnije formate . Za formate kao što su : PNG, TIFFF i JPEG2000 moguće je spremiti sliku kao 16 – bitnu ili četri – kanalnu (BGR plus alpha). Na mjesto drugog argumenta dolazi ime varijable u koja sadrži sliku koju želimo spremiti.
Funkcija će vratiti vrijednost 1 ako se uspješno izvrši ili vrijednost 0 ako se pojavi određena greška.

3.2 Segmentacija

Segmentacija je postupak kojim izoliramo određeni objekt ili dio objekta sa ostatka slike. U našem slučaju radi se o predmetu kojeg program mora prepoznati. Koristi će se metoda „oduzimanje pozadine“. Riječ je o jednostavnoj metodi koja uspoređuje trenutnu sliku sa naučenim modelom pozadine. Ono što ostane nakon usporedbe predstavlja traženi objekt. Model pozadine definiran je srednjom vrijednošću piksela kada ispod kamere nije postavljen nikakav predmet. Potrebno je definirati i određeno odstupanje. Za računanje graničnih vrijednosti koristi se prosječna promjena vrijednosti piksela između dva susjedna frame-a.



Slika 3-1 Segmentacija

Svi pikseli koji će se nalaziti unutar graničnih vrijednosti predstavljat će pozadinu. Pikseli čija je vrijednost veća ili manja od granične definirat će traženi predmet.

Najprije treba definirati sve potrebne varijable:

```
IplImage *IavgF,*IdiffF,*IprevF,*IhiF,*IlowF;

IplImage *Iscratch,*Iscratch2;

IplImage *Kanal1,*Kanal2,*Kanal3;
IplImage *Ilow1,*Ilow2,*Ilow3;
IplImage *Ihi1,*Ihi2,*Ihi3;

IplImage *Imaskt,*Imask;
float Icount;
```

Zatim slijedi alokacija svih potrebnih varijabli. Radi preglednosti program je razdijeljen na manje funkcije.

```
void AlokacijaSlika( CvSize size){

IavgF = cvCreateImage( size, IPL_DEPTH_32F, 3 );
            IdiffF = cvCreateImage( size, IPL_DEPTH_32F, 3 );
            IprevF = cvCreateImage( size, IPL_DEPTH_32F, 3 );
            IhiF = cvCreateImage( size, IPL_DEPTH_32F, 3 );
            IlowF = cvCreateImage(size, IPL_DEPTH_32F, 3 );
            Ilow1 = cvCreateImage( size, IPL_DEPTH_32F, 1 );
            Ilow2 = cvCreateImage( size, IPL_DEPTH_32F, 1 );
            Ilow3 = cvCreateImage( size, IPL_DEPTH_32F, 1 );
            Ihi1 = cvCreateImage( size, IPL_DEPTH_32F, 1 );
            Ihi2 = cvCreateImage( size, IPL_DEPTH_32F, 1 );
            Ihi3 = cvCreateImage( size, IPL_DEPTH_32F, 1 );
            cvZero(IavgF);
            cvZero(IdiffF);
            cvZero(IprevF);
            cvZero(IhiF);
            cvZero(IlowF);
            Icount = 0.00001;

      Iscratch = cvCreateImage( size, IPL_DEPTH_32F, 3 );
      Iscratch2 = cvCreateImage( size, IPL_DEPTH_32F, 3 );
      Kanal1 = cvCreateImage( size, IPL_DEPTH_32F, 1 );
      Kanal2 = cvCreateImage( size, IPL_DEPTH_32F, 1 );
      Kanal3 = cvCreateImage( size, IPL_DEPTH_32F, 1 );
      Imaskt = cvCreateImage( size, IPL_DEPTH_8U, 1 );
      cvZero(Iscratch);
      cvZero(Iscratch2);

}
```

Funkcijom akumulacijaPozadine( ) spremamo vrijednosti piksela odeđenog broja framova kako bi mogli definirati model pozadine. Funkcija traži samo jedan argument i to trenutni frame.

```
void akumulacijaPozadine(IplImage *frame)
{
      static int first = 1;
      cvCvtScale(frame,Iscratch,1,0); // float
      if (!first){
            cvAcc(Iscratch,IavgF);
            cvAbsDiff(Iscratch,IprevF,Iscratch2);
            cvAcc(Iscratch2,IdiffF);
            Icount += 1.0;
      }
      first = 0;
      cvCopy(Iscratch,IprevF);
}
```

Pri prvom zvanju funkcije akumulacijaPozadine( ) izvršava se samo funkcija cvCvtScale( ) i cvCopy( ) . Funkcija cvCvtScale( ) sprema trenutni frame u varijablu Iscratch, ali pritom mijenja rezoluciju slike iz 8 – bitne u 32- bitnu. Funkcija cvCopy( ) sprema vrijednost varijable Iscratch u varijablu IprevF koja služi za spremanje prethodnih framova.

Pri sljedećem zvanju funkcije koristi se funkcija cvAcc koja sprema vrijednosti piksela u varijablu IavgF. Funkcijom cvAbsDiff ( ) računamo apsolutnu razliku piksela između susjednih framova. Razlike spremamo u varijablu IdiffF ponovno funkcijom cvAcc( ). Na kraju povećavamo varijablu Icount za vrijednost 1, koju koristimo za računanje prosječne vrijednosti.

Nakon što smo spremili dovoljno framova, radimo model pozadine funkcijom stvaranjeModela( ).

```
void stvaranjeModela()
{
            cvConvertScale(IavgF,IavgF,(double)(1.0/Icount));

            cvConvertScale(IdiffF,IdiffF,(double)(1.0/Icount));

            cvAddS(IdiffF,cvScalar(1.0,1.0,1.0),IdiffF);
            gornjaGranica(4.0);
            donjaGranica(4.0);
      }
```

Računanje prosječne vrijednosti pojedinog piksela pozadine radimo funkcijom cvConvertScale( ).

Funkcija cvConvertScale( ) se sastoji od više različitih funkcija spojenih u jednu. Na primjer, ako je potrebno pretvoriti 8 bitnu RGB sliku u 16- bitnu možemo koristiti funkciju cvConvertScale( ). Funkcija također omogućava linearnu transformaciju podataka. Funkcija ne koristi operaciju dijeljenja nego množi vrijednosti pojedinih piksela sa konstantom koju definiramo kao treći argument. Zbog toga varijablu Icount moramo podijeliti sa jedinicom. Rezultat se sprema u varijablu koja se nalazi na mjestu drugog argumenta, u našem slučaju radi se o varijabli IavgF.

Prototip funkcije izgleda ovako:

```
void cvConvertScale(
     const CvArr* src,
     CvArr*       dst,
     double       scale = 1.0,
     double       shift = 0.0
     );
```

Pomoću četvrtog argumenta možemo još dodati određenu vrijednost pojedinom pikselu.

Funkcijom cvAddS dodajemo varijabli IdiffF konstantu 1. Na taj način se osiguravamo da granične vrijednosti ne budu jednake srednjoj vrijednosti pozadine. Prototip funkcije cvAddS izgleda ovako:

```
void cvAddS(
     const CvArr* src,
     CvScalar     value,
     CvArr*       dst,
     const CvArr* mask = NULL
);
```

Na kraju pozivamo funkcije gornja Granica ( ) i donjaGranica() kojima definiramo granične vrijednosti, time je svaranje modela pozadine završeno.

```
void gornjaGranica(float scale)
{
      cvConvertScale(IdiffF,Iscratch,scale);
      cvAdd(Iscratch,IavgF,IhiF);
      cvSplit( IhiF, Ihi1,Ihi2,Ihi3, 0 );
}


void donjaGranica(float scale)
{
      cvConvertScale(IdiffF,Iscratch,scale);
```

```
        cvSub(IavgF,Iscratch,IlowF);
        cvSplit( IlowF, Ilow1,Ilow2,Ilow3, 0 );
    }
```

U funkcijama gornjaGranica( ) i donjaGranica( ) ponovno koristimo funkciju cvCvtScale( ) kojom množimo prosječnu vrijednost razlike piksela susjednih framova sa određenom konstantom kako bi dobili gornju i donju graničnu vrijednost pozadine. Funkcijom cvSplit ( ) granične vrijednosti dijelimo na svaki kanal zasebno.

Nakon što smo napravili model pozadine zajedno sa graničnim vrijednostima možemo izvršiti postupak segmentacije. Funkcija razlikaPozadine( ) kojom vršimo postupak segmentacije traži dva argumenta. Na mjestu prvog argumenta je trenutni frame, a na drugom slika u koju spremamo rezultat.

```
void razlikaPozadine(IplImage *frame,IplImage *Imask)
{
    cvCvtScale(frame,Iscratch,1,0);
    //Kanal 1
    cvSplit( Iscratch, Kanal1,Kanal2,Kanal3, 0 );
    cvInRange(Kanal1,Ilow1,Ihi1,Imask);
    //Kanal 2
    cvInRange(Kanal2,Ilow2,Ihi2,Imaskt);
    cvOr(Imask,Imaskt,Imask);
    //Kanal 3
    cvInRange(Kanal3,Ilow3,Ihi3,Imaskt);
    cvOr(Imask,Imaskt,Imask);

    cvSubRS( Imask, cvScalar(255), Imask);
}
```

Naredbom cvCvtScale( ) trenutnu sliku kojoj je rezolucija 8 – bitna pretvaramo u privremenu sliku naziva Iscratch ali rezolucije 32 bita. Sliku zatim razdvajamo na pojedine kanale funkcijom cvSplit ( ).

Funkcijom cvInRange uspoređujemo svaki kanal sa gornjom i donjom graničnom vrijednošću te rezultat spremamo u sliku Imask. Ako je vrijednost piksela unutar graničnih vrijednosti funkcija će vrijednost istog piksela na slici Imask postaviti na 255 . Prototip funkcije izgleda ovako:

```
void cvInRange(
     const CvArr* src,
     const CvArr* lower,
     const CvArr* upper,
     CvArr*       dst
);
```

Funkcija cvOr() radi logičku operaciju ILI, njome dobivene rezultate za pojedini kanal spremamo u sliku Imask. Na kraju je potrebno invertirati dobiveni rezultat funkcijom cvSubRS ( ), jer traženi predmet predstavljaju pikseli kojima je vrijednost izvan graničnog područja.

Na kraju čistimo memoriju funkcijom dealokacijaSlika ( ).

```
void DealokacijaSlika()
{

    cvReleaseImage(&IavgF);
    cvReleaseImage(&IdiffF );
```

```
                cvReleaseImage(&IprevF );
                cvReleaseImage(&IhiF );
                cvReleaseImage(&IlowF );
                cvReleaseImage(&Ilow1 );
                cvReleaseImage(&Ilow2  );
                cvReleaseImage(&Ilow3 );
                cvReleaseImage(&Ihi1   );
                cvReleaseImage(&Ihi2   );
                cvReleaseImage(&Ihi3  );

          cvReleaseImage(&Iscratch);
          cvReleaseImage(&Iscratch2);

          cvReleaseImage(&Kanal1  );
          cvReleaseImage(&Kanal2 );
          cvReleaseImage(&Kanal3 );

          cvReleaseImage(&Imaskt);
          cvReleaseImage(&Imask);
     }
```

Pri razvoju programa primjećeno je da kameri treba određeno vrijeme za prikazivnje čiste slike. Zbog toga je u program dodana određena logika pomoću koje funkciju akumulacijaPozadine( ) zovemo nakon 20 – tog frame –a. Funkcija akumulacijaPozadine poziva se još pet puta kako bi dobili dovoljnu količinu spremljenih frame- ova. Nakon toga poziva se funkcija stvaranjeModela( ) . Kada je stvoren model pozadine može se koristiti funkicja razlikaPozadine( )kojom radimo segmentaciju.

```
if( k >=20 && k < 26)
            {
                  akumulacijaPozadine(frame);
                  k++;

                  if(k == 25 )
                  { printf("\n pozadina naucena");
                    stvaranjeModela();
                  }
          }

 if(k == 26 )
                {
                      razlikaPozadine(frame,Imask);
                      cvSmooth( Imask, Imask, CV_GAUSSIAN, 3, 3
);
                }


                           .
                           .
                           .


if(k <=20)
            {k++;}
```

3.3 Raspoznavanje predmeta

Predmete prije raspoznavanja spremamo u bazu pomoću funkcije zapamtiPredmete(). Funkcija kao argument treba binarnu sliku koju smo dobili postupkom segmentacije. Pomoću funkcije cvCountNonZero( ) računamo broj bijelih piksela na slici, odnosno površinu predmeta. Informacija o površini predmeta sprema se u polje naziva broj_piksela. U spomenuto polje možemo spremiti do četri predmeta. Na kraju se funkcijom printf () ispisuje broj piskela te obavijest da je predmet spremljen.

```
int b=0;
 unsigned int broj_piksela[] = {0, 0, 0, 0};

 void zapamtiPredmet(IplImage *Imask){
       broj_piksela[b] = cvCountNonZero( Imask);
       printf("\n Broj piksela: %d", broj_piksela[b]);
       b++;
     printf("\n  Predmet spremljen");
     }
```

Brisanje spremljenih predmeta vršimo pomoću funkcije brisanje ( ). Funkcija postavlja članove polja na nulu te ispisuje poruku da su predmeti izbrisani.

```
void brisanje (){
       b=0;
       unsigned int broj_piksela[]= {0,0,0,0};
       printf("\n Predmeti izbrisani");
       }
```

Uspoređivanje vršimo pomoću funkcije usporedivanje( ) koja se poziva pritiskom na tipku „ u “. Funkcija najprije računa broj piksela predmeta koji se nalazi ispod kamere, zatim dobiveni broj piksela uspoređuje sa spremljenim vrijednostima. Dopušteno je odstupanje u broju piksela do 10 %. Kod računanja dozvoljenog odstupanja može se pojaviti racionalni broj, zbog toga se koristi funkcija cvRound () koja pretvara racionalni broj (float) u cjelobrojni tip podataka (int).

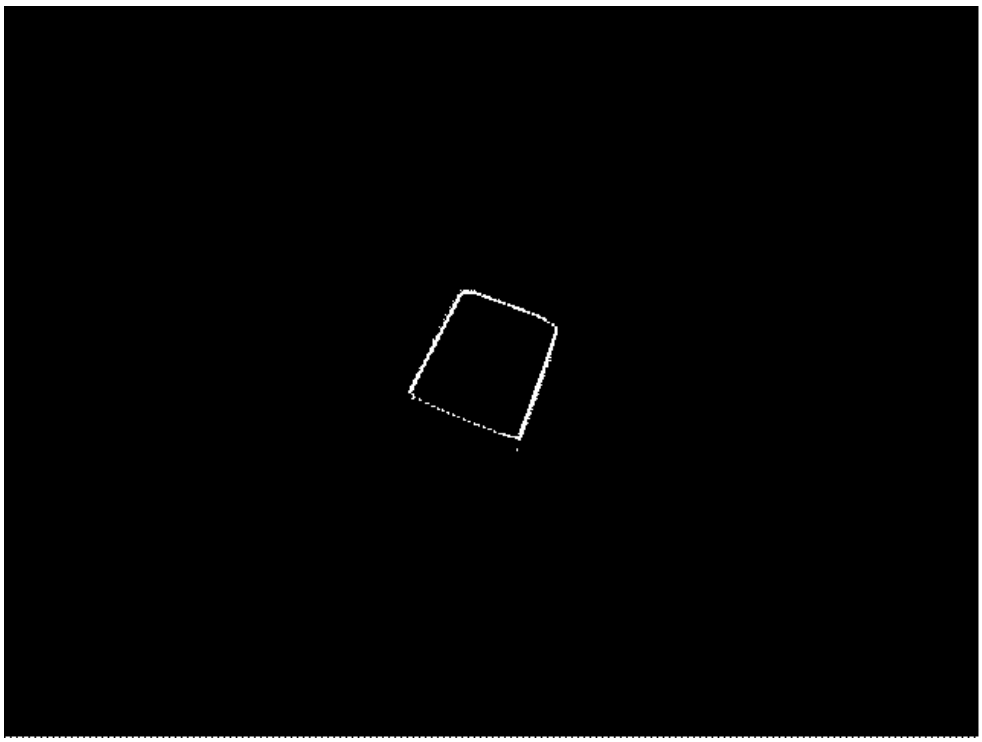
```
void usporedivanje(IplImage *Imask){

   unsigned int tren_broj;
   int z = 0;

     tren_broj = cvCountNonZero(Imask);

      for ( z = 0; z < b; z++)
      {
         if ( broj_piksela[z] <= cvRound(tren_broj/10) + tren_broj &&
broj_piksela[z] >= tren_broj - cvRound(tren_broj/10) )
            {
                     if ( z == 0)
                           printf("\n Predmet A ");
                     if ( z == 1)
                           printf("\n Predmet B ");
                     if ( z == 2)
                           printf("\n Predmet C");
            }
       }
```

3.4 Primjer rada programa

Prije detektiranja konture predmeta slika (Slika 3-3) potrebno je definirati graničnu vrijednost susjednih piksela (Slika 3-2).

Slika 3-2 Definiranje razlike između piksela

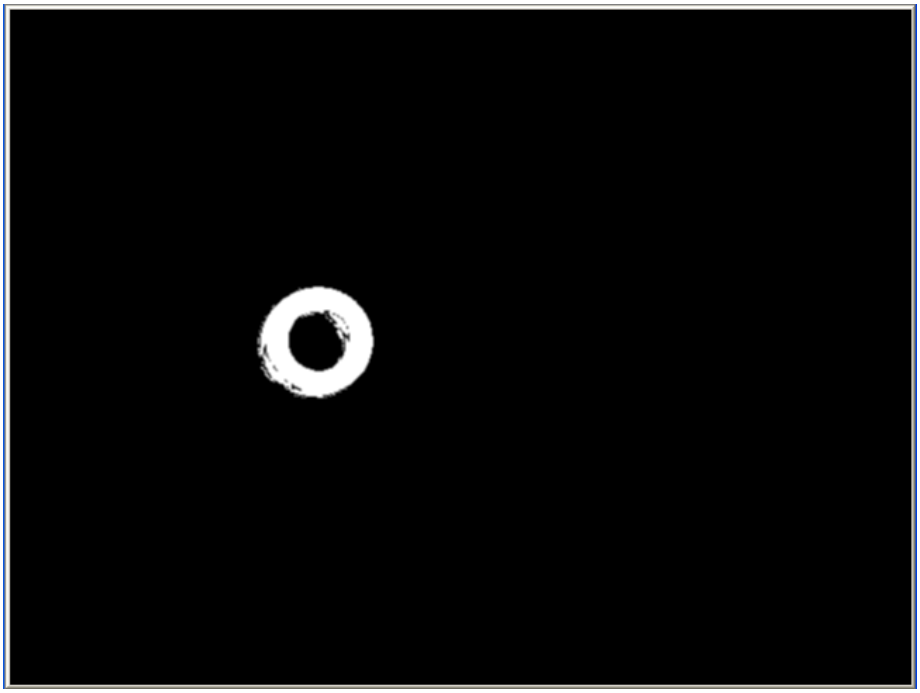
Slika 3-3 Predmet kojemu tražimo konturu



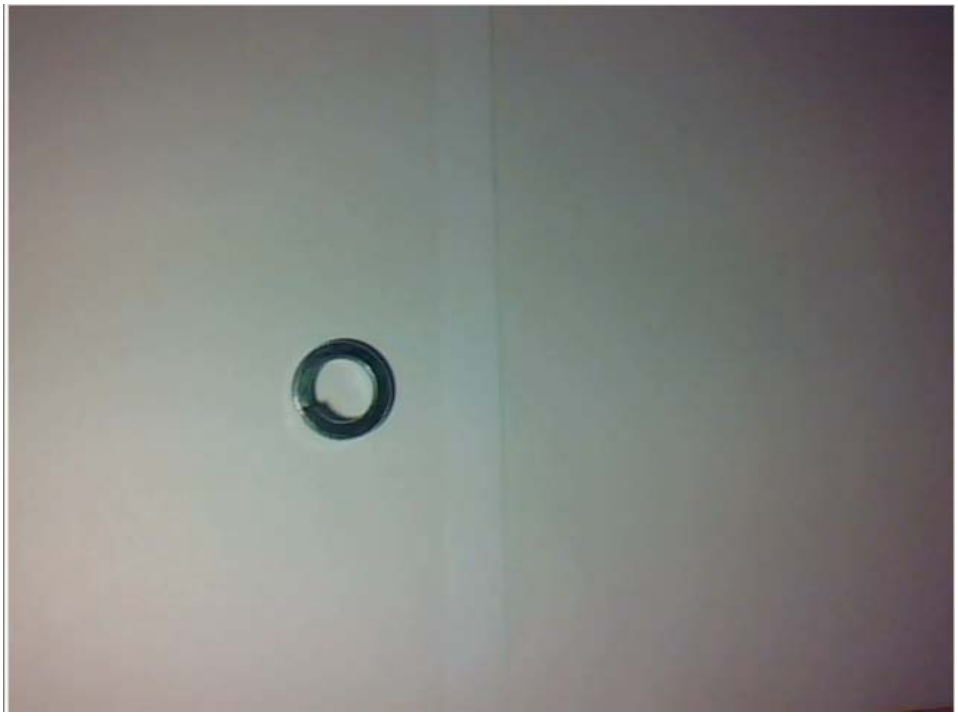
Slika 3-4 Dobivena kontura predmeta

Prije prepoznavanja predmeta potrebno je spremiti njihove površine. Na donjim slikama prikazan je postupak spremanja površina predmeta. Površina predmeta je iskazana u broju piksela.

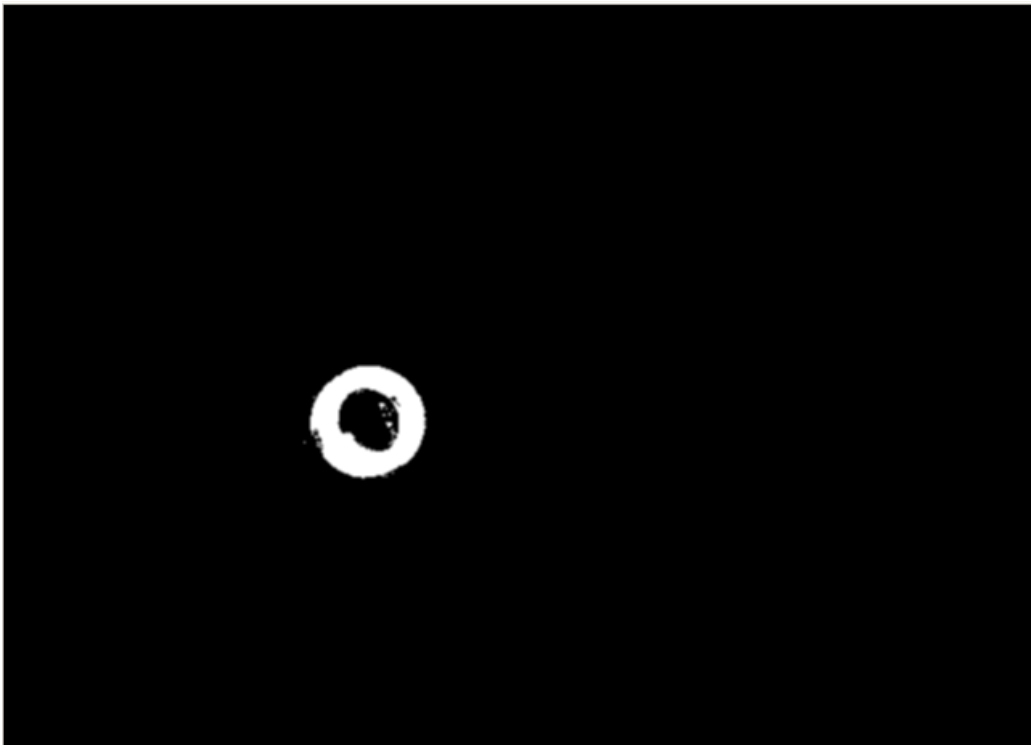
Slika 3-5 Predmet A



Slika 3-6 Predmet A: segmentacija



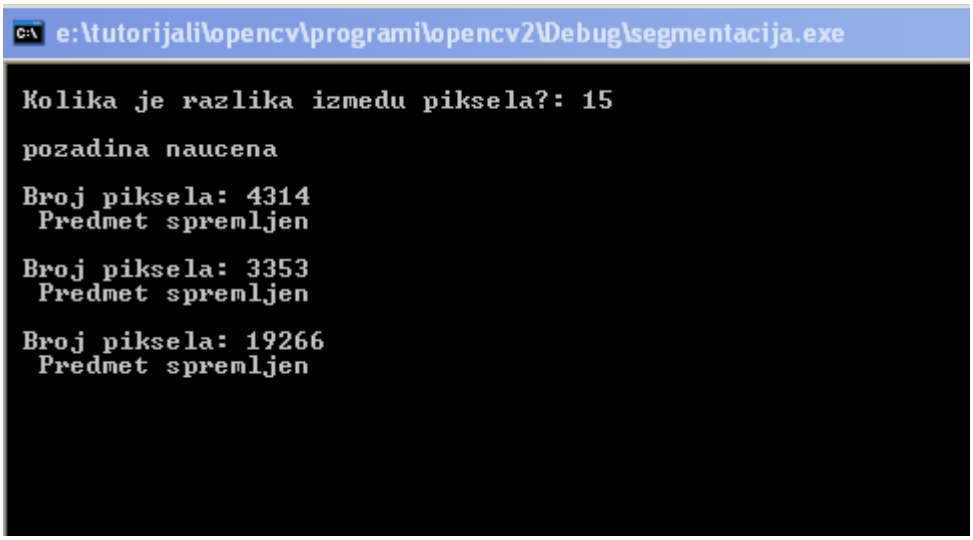
Slika 3-7 Predmet B



Slika 3-8 Predmet B : segmentacija

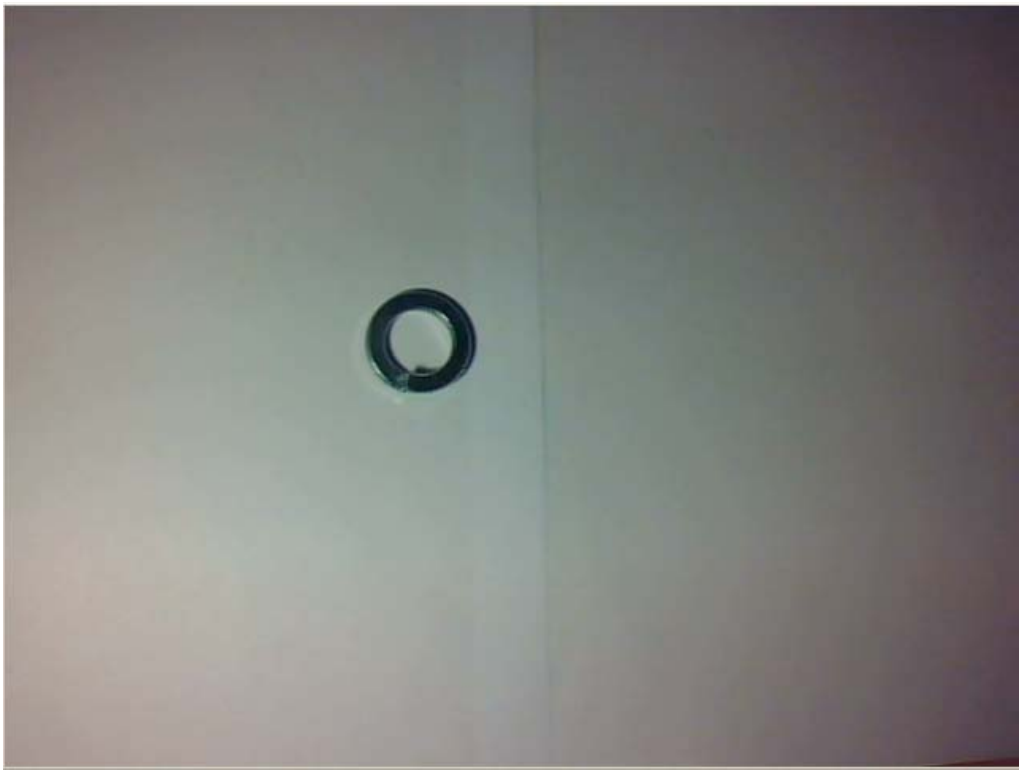
Slika 3-9 Predmet C

Slika 3-10 Predmet C: segmentacija

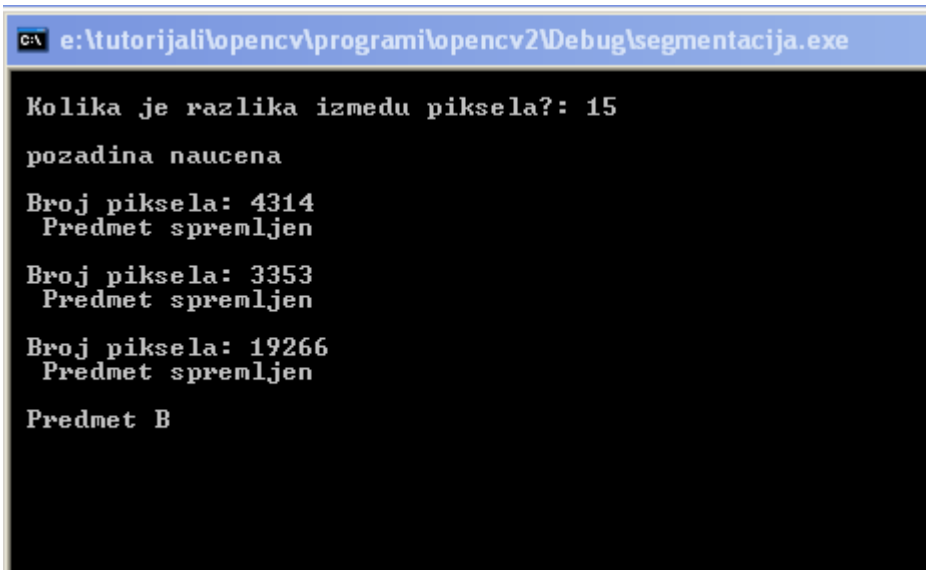


Slika 3-11 Ispis broja piksela za pojedine predmete

Nakon što smo spremili predmete slijedi prepoznavanje. Prepoznavanje predmeta vršimo pritiskom na tipku „ u “. Na donjim slikam prikazan je primjer prepoznavanja spremljenih predmeta.

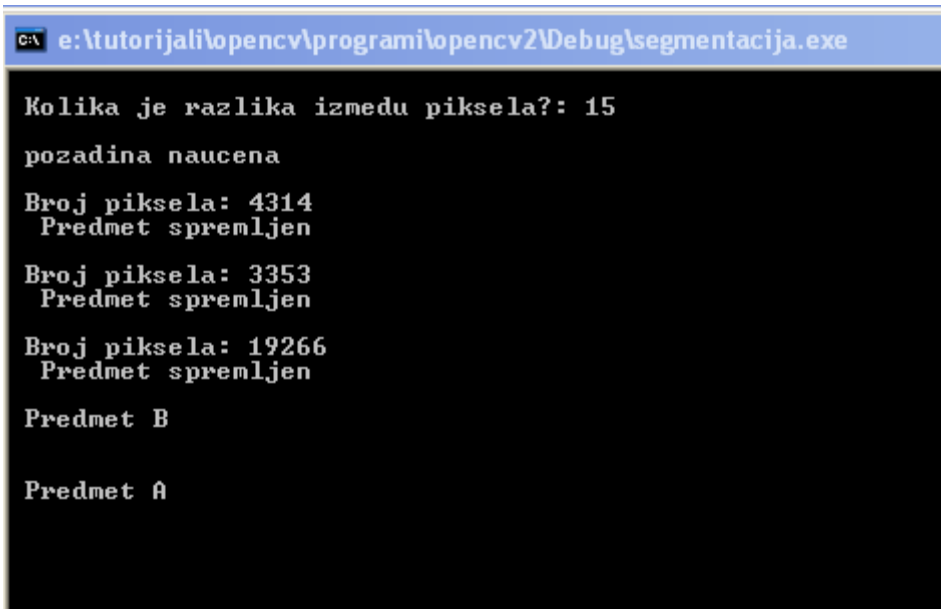


Slika 3-12 Prepoznavanje predmeta (Predmet B)



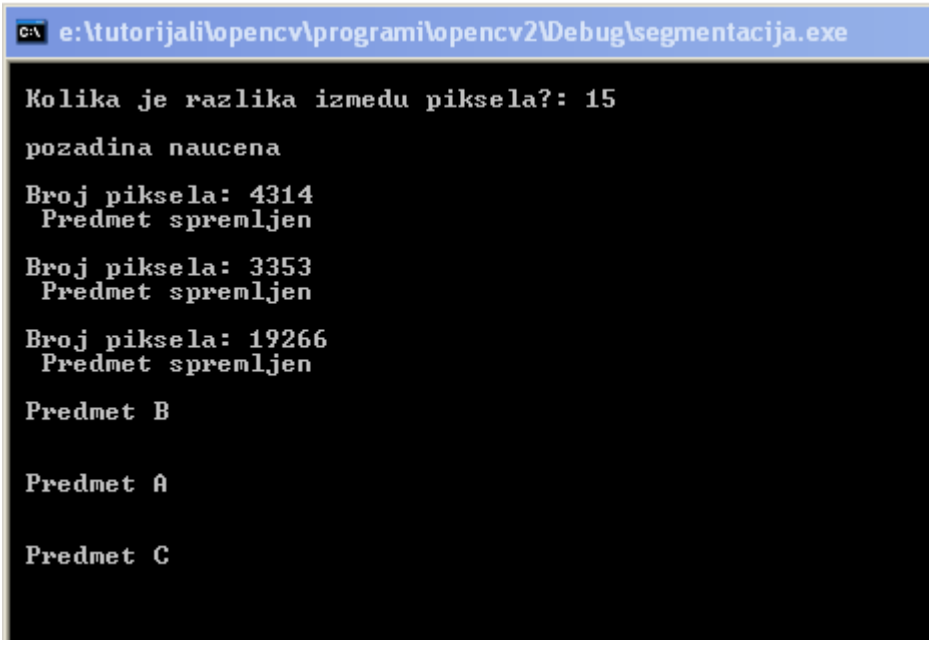
Slika 3-13 Uspješno prepoznat predmet B

Slika 3-14 Prepoznavanje predmeta (Predmet A)



Slika 3-15 Uspješno prepoznat predmet A

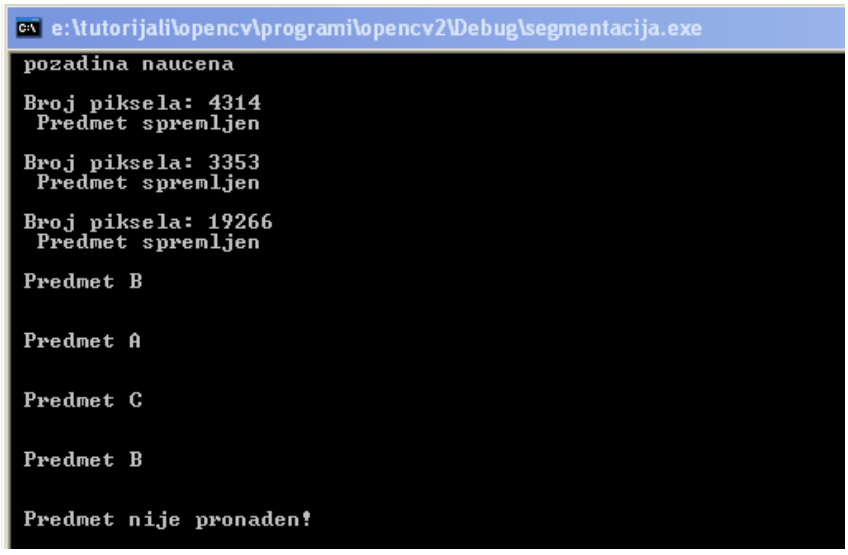
Slika 3-16 Prepoznavanje predmeta (Predmet C)



Slika 3-17 Uspješno prepoznat predmet C

Nedostatak programa je što nije u mogućnosti prepoznati predmete jednake površine. Loše osvjetljenje također radi probleme. Na donjoj slici je prikazano neuspješno prepoznavanje predmeta B

Slika 3-18 Prepoznavanje predmeta (Predmet B)



Slika 3-19 Neuspješno prepoznavanje predmeta B

4. Zaključak

Zadatak je bio pomoću biblioteke OpenCV napisati program za otkrivanje kontura objekta primjenom metode značajnih razlika. Algoritam je pokazao solidne rezultate kod otkrivanja kontura predmeta. Najveći problem mu predstavlja neodgovarajuće osvjetljenje predmeta jer se može dogoditi da program detektira sjenu kao konturu predmeta. Za složenije aplikacije bolje je koristiti već gotove algoritme koji dolaze sa bibliotekom OpenCV, primjerice funkcija cvFindContours.

Program je proširen kako bi mogao raspoznavati objekte (eng. object recognition). Značajka kojom se vršilo prepoznavanja predmeta bila je površina predmeta odnosno broj piksela bijele boje na binarnoj slici dobivene segmentacijom. Riječ je od jednostavnoj metodi koja je pokazala solidne rezultate. Najveća joj je mana što nije u mogućnosti prepoznati predmete jednake površine.

Za pisanje programa odabrao sam programski jezik C. Sljedeći put radije ću uzeti programski jezik Python nego C ili C++. Python je interpreterski, interaktivni, objektni programski jezik. Riječ je o besplatnom (open – source) jeziku, sa dobrom literaturom te mnoštvo podržanih biblioteka uključujući i OpenCV. Pisanje koda je vrlo jednostavno u odnosu na C/C++ obzirom da je Python viši programski jezik od C/C++ . Programi se mogu izvršavati nešto dulje ali kod jednostavnijih aplikacija to ne bi trebao biti problem.

Python također omogućava jednostavniju izradu grafičkog sučelja. Sve potrebne informacije vezane za program prikazivale bi se u grafičkom sučelju što je daleko bolje rješenje nego korištenje komandnog prozora. Prednost Pythona je i u podržavanju ostalih operativnih sustava kao što je MaxOS i Linux. Uz vrlo male izmjene program može raditi na Linuxu koji je besplatan. Na taj način za potrebe rada aplikacije ne bi bio potreban niti jedan komercijalni program ili operativni sustav, jedino bi se trebao kupiti odgovarajući hardver.

Literatura

[1] Bradski G., Kaehler A. : Learning OpenCV: Computer Vision with the OpenCV Library, O'Reilly Media , Sebastopol CA, 2008.

[2] Machine vision – Wikipedia , http://en.wikipedia.org/wiki/Machine_vision , 25. siječanj 2010.

[3] Computer vision – Wikipedia , http://en.wikipedia.org/wiki/Computer_vision , 25. siječanj 2010.

[4] OpenCV_1.1pre1a.exe, OpenCV library, http://sourceforge.net/project/showfiles.php?group_id=22870&package_id=16937, 25. listopada 2009.

[5] Yahoo group OpenCV : http://tech.groups.yahoo.com/group/OpenCV/ , 1. prosinac 2009.

[6] Welcome - OpenCV Wiki , http://opencv.willowgarage.com/wiki/ , 10. studeni 2009.

[7] Blog – OpenCV , http://myopencv.wordpress.com/, 10. studeni 2009.

.

Prilog: programski kod

```
#include <cv.h>
#include <cxcore.h>
#include <highgui.h>
#include <stdio.h>


// spremanje predmeta

 int b=0,z=0;
 unsigned int broj_piksela[] = {0, 0, 0, 0};

 void zapamtiPredmet(IplImage *Imask){
       broj_piksela[b] = cvCountNonZero( Imask);
       printf("\n Broj piksela: %d", broj_piksela[b]);
       b++;
     printf("\n  Predmet spremljen");
       printf("\n");
     }

 // brisanje predmeta

 void brisanje (){
       b=0;
       unsigned int broj_piksela[]= {0,0,0,0};
       printf("\n Predmeti izbrisani");
       printf("\n");
       }


       // uspoređivanje predmeta

       unsigned int tren_broj;

       void usporedivanje(IplImage *Imask){
       int x=0;
       tren_broj = cvCountNonZero(Imask);

      for ( z = 0; z < b; z++)
      {
         if  (  broj_piksela[z]  <=  cvRound(tren_broj/10)  +  tren_broj  &&
broj_piksela[z] >= tren_broj - cvRound(tren_broj/10) )
             {
                         if ( z == 0)
                               printf("\n Predmet A  ");

                              x++;
                         if ( z == 1)
                               printf("\n Predmet B ");

                             x++;
                         if ( z == 2)
                               printf("\n Predmet C ");

                             x++;
             }

        }
        if(x==0)
                   printf("\n Predmet nije pronaden!");
                 printf("\n");

       }
```

```
    // segmentacija

IplImage *IavgF,*IdiffF,*IprevF,*IhiF,*IlowF;

IplImage *Iscratch,*Iscratch2;

IplImage *Kanal1,*Kanal2,*Kanal3;
IplImage *Ilow1,*Ilow2,*Ilow3;
IplImage *Ihi1,*Ihi2,*Ihi3;

IplImage *Imaskt,*Imask;
float Icount;
CvSize size;


void AlokacijaSlika( CvSize size){

IavgF = cvCreateImage( size, IPL_DEPTH_32F, 3 );
            IdiffF = cvCreateImage( size, IPL_DEPTH_32F, 3 );
            IprevF = cvCreateImage( size, IPL_DEPTH_32F, 3 );
            IhiF = cvCreateImage( size, IPL_DEPTH_32F, 3 );
            IlowF = cvCreateImage(size, IPL_DEPTH_32F, 3 );
            Ilow1 = cvCreateImage( size, IPL_DEPTH_32F, 1 );
            Ilow2 = cvCreateImage( size, IPL_DEPTH_32F, 1 );
            Ilow3 = cvCreateImage( size, IPL_DEPTH_32F, 1 );
            Ihi1 = cvCreateImage( size, IPL_DEPTH_32F, 1 );
            Ihi2 = cvCreateImage( size, IPL_DEPTH_32F, 1 );
            Ihi3 = cvCreateImage( size, IPL_DEPTH_32F, 1 );
            cvZero(IavgF);
            cvZero(IdiffF);
            cvZero(IprevF);
            cvZero(IhiF);
            cvZero(IlowF);
            Icount = 0.00001; //dijeljenje sa nulom

      Iscratch = cvCreateImage( size, IPL_DEPTH_32F, 3 );
      Iscratch2 = cvCreateImage( size, IPL_DEPTH_32F, 3 );
      Kanal1 = cvCreateImage( size, IPL_DEPTH_32F, 1 );
      Kanal2 = cvCreateImage( size, IPL_DEPTH_32F, 1 );
      Kanal3 = cvCreateImage( size, IPL_DEPTH_32F, 1 );
      Imaskt = cvCreateImage( size, IPL_DEPTH_8U, 1 );
      cvZero(Iscratch);
      cvZero(Iscratch2);

}

void gornjaGranica(float konstanta)
{
      cvConvertScale(IdiffF,Iscratch,konstanta);
      cvAdd(Iscratch,IavgF,IhiF);
      cvSplit( IhiF, Ihi1,Ihi2,Ihi3, 0 );
}


void donjaGranica(float konstanta)
{
      cvConvertScale(IdiffF,Iscratch,konstanta);
      cvSub(IavgF,Iscratch,IlowF);
      cvSplit( IlowF, Ilow1,Ilow2,Ilow3, 0 );
}


void stvaranjeModela()
{
```

```
            cvConvertScale(IavgF,IavgF,(1.0/Icount));   // srednja vrijednost
piksela
            cvConvertScale(IdiffF,IdiffF,(1.0/Icount)); // srednja vrijednost
razlike
            cvAddS(IdiffF,cvScalar(1.0,1.0,1.0),IdiffF);  //diff najmanje 1
            gornjaGranica(4.0);
            donjaGranica(4.0);
}


void razlikaPozadine(IplImage *frame,IplImage *Imask)
{
      cvCvtScale(frame,Iscratch,1,0);
      //Kanal 1
      cvSplit( Iscratch, Kanal1,Kanal2,Kanal3, 0 );
      cvInRange(Kanal1,Ilow1,Ihi1,Imask);
      //Kanal 2
      cvInRange(Kanal2,Ilow2,Ihi2,Imaskt);
      cvOr(Imask,Imaskt,Imask);
      //Kanal 3
      cvInRange(Kanal3,Ilow3,Ihi3,Imaskt);
      cvOr(Imask,Imaskt,Imask);

      cvSubRS( Imask, cvScalar(255), Imask);
}
void akumulacijaPozadine(IplImage *frame)
{
      static int prvi = 1;
      cvCvtScale(frame,Iscratch,1,0); // float
      if (!prvi){
            cvAcc(Iscratch,IavgF);
            cvAbsDiff(Iscratch,IprevF,Iscratch2);
            cvAcc(Iscratch2,IdiffF);
            Icount += 1.0;
      }
      prvi = 0;
      cvCopy(Iscratch,IprevF);
}

void DealokacijaSlika()
{

            cvReleaseImage(&IavgF);
            cvReleaseImage(&IdiffF );
            cvReleaseImage(&IprevF );
            cvReleaseImage(&IhiF );
            cvReleaseImage(&IlowF );
            cvReleaseImage(&Ilow1 );
            cvReleaseImage(&Ilow2  );
            cvReleaseImage(&Ilow3 );
            cvReleaseImage(&Ihi1   );
            cvReleaseImage(&Ihi2   );
            cvReleaseImage(&Ihi3  );

      cvReleaseImage(&Iscratch);
      cvReleaseImage(&Iscratch2);

      cvReleaseImage(&Kanal1  );
      cvReleaseImage(&Kanal2 );
      cvReleaseImage(&Kanal3 );

      cvReleaseImage(&Imaskt);
      cvReleaseImage(&Imask);
```

```
}


int main( int argc, char** argv ) {


int height,width,step,channels;    //varijable za spremanje rezolucije videa,
broj kanala

int plava,zelena,crvena;     // varijable u koje se sprema vrijednost boje za
pojedini piksel

int i=0,j=0,k=0,a=0;
                        // for petlja, i - visina videa (height) j - širina
videa( width )

int stepr, channelsr;  //
int diff;           // konstanta kojom definiramo  razliku između piksela

uchar *data,*datar;

int plava_plus,zelena_plus,crvena_plus;   // varijable za spremanje graničnih
vrijednosti pojedinih boja
int plava_minus,zelena_minus,crvena_minus;


CvMemStorage* storage = cvCreateMemStorage(0);

char c;

printf("\n Kolika je razlika izmedu piksela?: ");
scanf("%d",&diff);


    CvCapture* capture = cvCreateCameraCapture(0);

    CvSize    size    =    cvSize(    (int)cvGetCaptureProperty(    capture,
CV_CAP_PROP_FRAME_WIDTH),
                          (int)cvGetCaptureProperty(                capture,
CV_CAP_PROP_FRAME_HEIGHT));


   IplImage* rezultat  = cvCreateImage( size,IPL_DEPTH_8U, 1);
   IplImage* siva   = cvCreateImage( size,IPL_DEPTH_8U, 1);
             Imask = cvCreateImage( size, IPL_DEPTH_8U, 1 );
   IplImage* foreground  = cvCreateImage( size,IPL_DEPTH_8U, 1);
   cvZero(foreground);
   cvZero(Imask);

      AlokacijaSlika(size);

   cvNamedWindow( "Kamera", CV_WINDOW_AUTOSIZE );
   cvNamedWindow( "Kontura", CV_WINDOW_AUTOSIZE );
   cvNamedWindow( "Segmentacija", CV_WINDOW_AUTOSIZE );

IplImage* frame;

while(1) {

        frame = cvQueryFrame( capture );
        if( !frame ) break;

             height = frame->height;
             width = frame->width;
```

```
        step =frame->widthStep;
        channels = frame->nChannels;
        data = (uchar *)frame->imageData;

        stepr=rezultat->widthStep;
        channelsr=rezultat->nChannels;
        datar = (uchar *)rezultat->imageData;

    for(i=0;i< (height-1);i++)

    {

         for(j=0;j<(width-1);j++)
             {
         plava = data[i*step+j*channels]; //plavi piksel trenutna vrijednost
         zelena = data[i*step+j*channels+1];
         crvena = data[i*step+j*channels+2];

         plava_plus = data[(i+1)*step+(j+1)*channels]+diff;
         zelena_plus = data[(i+1)*step+(j+1)*channels+1]+diff;
             crvena_plus = data[(i+1)*step+(j+1)*channels+2]+diff;

         plava_minus = data[(i+1)*step+(j+1)*channels]-diff;
         zelena_minus = data[(i+1)*step+(j+1)*channels+1]-diff;
         crvena_minus = data[(i+1)*step+(j+1)*channels+2]-diff;

if( (plava > plava_plus || plava < plava_minus )

&&( zelena > zelena_plus || zelena < zelena_minus )

&&( crvena > crvena_plus || crvena < crvena_minus ))

{
     datar[(i)*stepr+(j)*channelsr]=255;
}

 else datar[(i)*stepr+(j)*channelsr]=0;

     }
 }


      // segmentacija srednja vrijednost
      if( k >=20 && k < 25)
          {
                akumulacijaPozadine(frame);
                k++;

                if(k == 25 )
                { printf("\n pozadina naucena");
                printf("\n");
                  stvaranjeModela();
                }
        }

     if(k == 25 )
               {
                     razlikaPozadine(frame,Imask);
                     cvSmooth( Imask, Imask, CV_GAUSSIAN, 3, 3 );
               }



     cvShowImage( "Kamera", frame );
```

```
        cvShowImage( "Kontura", rezultat);
        cvShowImage( "Segmentacija", Imask );

        c = cvWaitKey(30);

                        if(c == 27 )
                              break;

          switch(c)
                              {
                                    case 's':

                                          zapamtiPredmet(Imask);

                                          cvWaitKey(100);
                                          break;

                                    case 'u':
                                          usporedivanje(Imask);

                                          break;
                                    case 'b':

                                          brisanje();
                                          break;

                                    }
            if(k <=20)
            {k++;}

            }

      cvSaveImage("kontura.jpg",rezultat);
      cvSaveImage("kamera.jpg",frame);
      cvSaveImage("slika.jpg",Imask);
      cvReleaseImage( &rezultat);
      cvReleaseCapture( &capture );
      DealokacijaSlika();
      cvDestroyAllWindows();

}
```