Sveučilište u Zagrebu
Fakultet strojarstva i brodogradnje

ZAVRŠNI RAD

Voditelj rada:

Prof. dr. sc. Mladen Crneković

Stipe Skroče

035172739

Zagreb

13. rujna 2013.

Izjava

Izjavljujem da sam ovaj rad radio samostalno, služeći se znanjem stečenim tijekom studija i koristeći navedenu literaturu.

Stipe Skroče 13. rujna 2013.

Sadržaj

Popis slika

SVEUČILIŠTE U ZAGREBU

FAKULTET STROJARSTVA I BRODOGRADNJE

Središnje povjerenstvo za završne i diplomske ispite

Povjerenstvo za završne ispite studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo materijala i mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa:	
Ur.broj:	

ZAVRŠNI ZADATAK

Student: **STIPE SKROČE** Mat. br.:0035172739

Naslov rada na hrvatskom jeziku: **PREPOZNAVANJE eMIR MOBILNIH ROBOTA KAMEROM**

Naslov rada na engleskom jeziku: **RECOGNITION OF eMIR MOBILE ROBOTS BY CAMERA**

Opis zadatka:

Tri eMIR mobilna robota (žuti, plavi i crveni) gibaju se na poligonu iznad kojega je postavljena kamera. Za potrebe vođenja robota potrebno je ostvariti prepoznavanje svakoga od njih, a podatke o prepoznatome serijskom komunikacijom slati na drugo računalo.
U radu je potrebno:

- ostvariti funkciju prepoznavanja tri robota zadane boje,
- informaciju o prepoznatom (boja, položaj i orijentacija) slati na drugo računalo.

Zadatak zadan:

16. studenog 2012.

Zadatak zadao:

Prof.dr.sc. Mladen Crneković

Rok predaje rada:
1. rok: 15. veljače 2013.
2. rok: 11. srpnja 2013.
3. rok: 13. rujna 2013.

Predviđeni datumi obrane:
1. rok: 27., 28. veljače i 1. ožujka 2013.
2. rok: 15., 16. i 17. srpnja 2013.
3. rok: 18., 19., i 20. rujna 2013.

Predsjednik Povjerenstva:

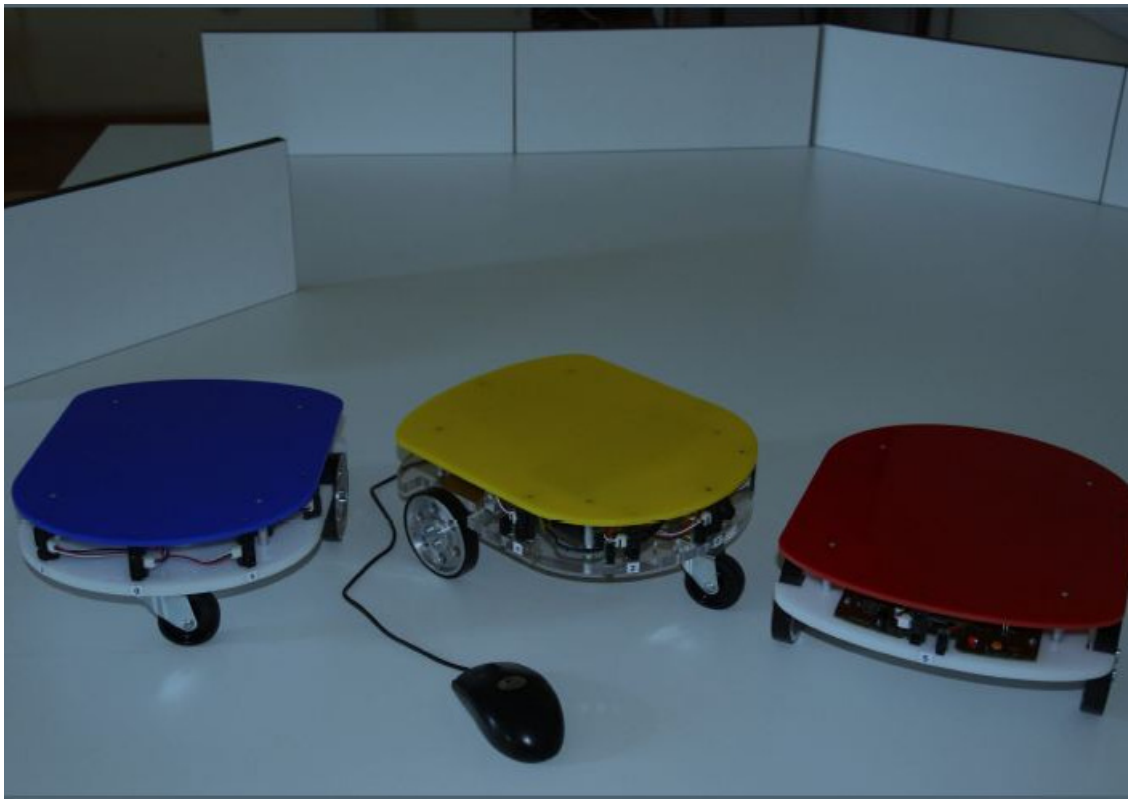
Prof. dr. sc. Zoran Kunica

1 Uvod

U ovome završnom je ostvariti funkciju prepoznavanja tri robota određene boje. Te informaciju o prepoznatom(boja, položaj i orjentacija) slati na drugo računalo. Iznad poligona se na visini od 3 metra nalazi kamera koja obuhvaća cijeli poligon. Na poligonu se nalaze tri oblikom ista robota. Razlikuju se samo u boji gornjeg sloja. Roboti se gibaju po poligonu te pritom jedno računalo prati preko kamere njihovu lokaciju i gibanje. Ti podaci se šalju serijskom vezom na računalo koje upravlja tim robotima. U ovom radu koristit će se programski jezik Python i Open Source Library od kojih je najvažniji OpenCV. Posebno zanimljiv aspekt ovog rada bit će napraviti program koji neće utjecati na rad robota zbog duljine procesa obrade slike, jer ako će kamera iznad poligona biti glavni senzor, onda mora obrada slike biti kontinuirana i brza.

2 e-MIR robot

e-MIR roboti izrađeni su ponajprije sa zadačom za izradu istraživanja u kognitivnoj robotici, programiranju robota te percepcije radnog okruženja robota. Na slici 1 nalazi se prikaz e-MIR robota na poligonu.



Slika 1: Prikaz e-MIR robota

Opremljeni su infracrvenim senzorima za udaljenost te kamerom. Platforma za programiranje ovih robota nije definirana te je izrazito fleksibilna u odabiru programskog jezika.

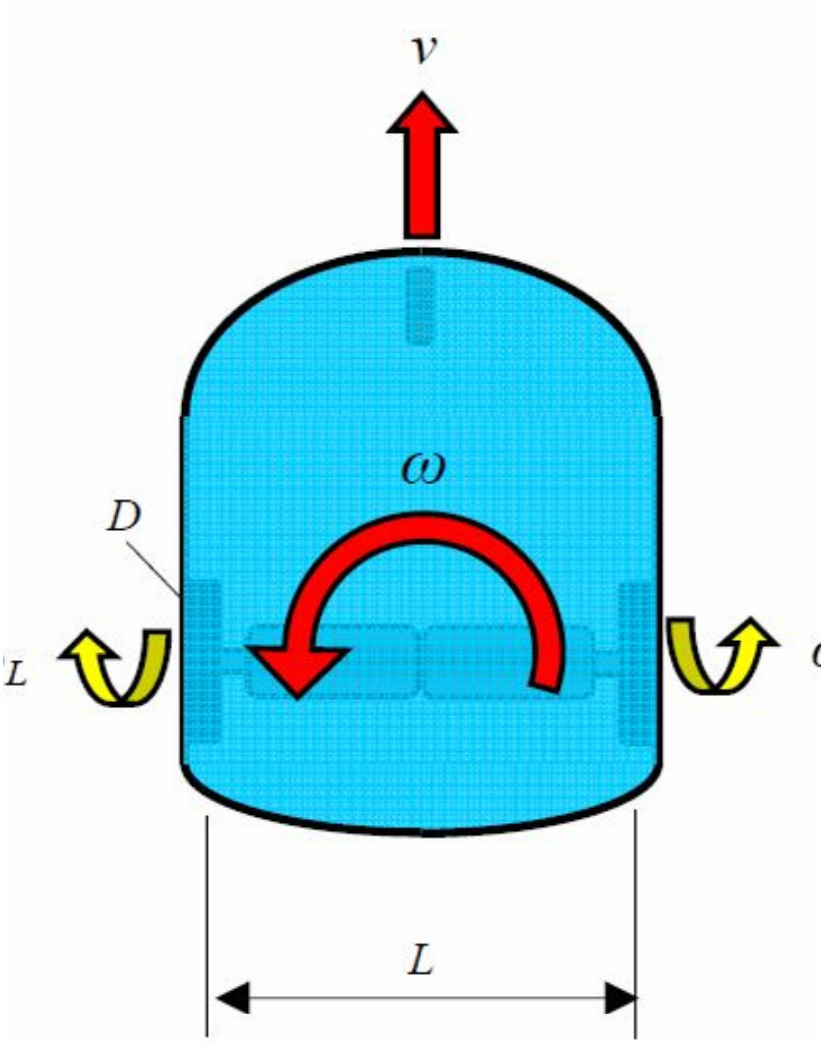
Jednađbe (1) i (2) opisuju kretanje robota:

$$v = \frac{D(\omega_R + \omega_L)}{4} \tag{1}$$

$$\omega = \frac{D(\omega_R - \omega_L}{2L} \tag{2}$$

Vrijednosti robota na slici 2 su sljedeče:

- L=240mm
- v_max=0.5m/s
- D=80mm(promjer kotača)



Slika 2: Dimenzije robota

Rad je strukturiran kako slijedi:

▷ U poglavlju 3. se pobliže objašnjava Računalni vid

▷ U poglavlju 4. se prezentira program Python i posebni library-ji s kojima smo ostvarili rješenje

▷ U poglavlju 5. dan je opsežan opis rada te su istaknuti i objašnjeni posebno značajni djelovi koda.

▷ U poglavlju 6. problematika Open Source programa

▷ U poglavlju 7. je zaključak.

▷ U poglavlju 8. je sažetak.

▷ Na kraju završnog rada dodan je cijeli kod u Python-u

3 Računalni vid

Računalni vid je umjetni sustav koji dobiva informacije obradom slika, te je jedan od važnijih disciplina u području umjetne inteligencije koja se bavi izradom sustava čiji je cilj izvlačenje informacija iz vizualnih izvora. Kako je većina informacija koje čovjek dobiva iz svoje okoline upravo vizualnog tipa, odmah postaje jasno da se pomoću računalnog vida također može priskrbljivati najviše informacija iz okoline. Računalni vid se razvio relativno kasno kao računarska disciplina, prvenstveno zbog ograničenosti procesorske moći računala, te su se tek u zadnje vrijeme tehnike iz ovog područja počele češće primjenjivati u praktične svrhe. Za sada se nije uspio stvoriti sustav koji uspješno nadomješta čovjekovu percepciju u cjelin, ali su se uspjeli izraditi specijalizirani sustavi koji u određenim usko specijaliziranim područjima ponekad čak i nadmašuju čovjeka (pronalaženje tumora na medicinskim slikama, prepoznavanje i rekonstruiranje oštećenih slika i slično). Praćenje objekata je proces određivanja položaja jednog ili više pokretnih objekata u vremenu. Zadatak algoritama za praćenje je analiza slikovnih sekvenci u svrhu izdvajanja korisne informacije. Praćenje osoba, vozila ili drugih objekata predstavlja važan aspekt današnjih i budućih aplikacija te stoga ima vrlo široku primjenu u praksi. Danas su poznati mnogi algoritmi čija je svrha praćenje objekata u slikovnom zapisu.[2]
Mogu se podijeliti na:

- Region based algoritmi su generički algoritmi koji koriste informaciju o boji i teksturi objekta kojeg prate.
- Contour based algoritmi izdvajaju i prate isključivo konture objekta.
- Generičko rješenje problema praćenja je još uvijek vrlo zahtjevan problem stoga se koriste i metode praćenja temeljene na modelu (model-based algoritmi) koje zahtjevaju apriorne informacije o obliku i tipu objekta.

4 Korišteni alati

4.1 Python

Python je objektno orjentirani programski jezik, kojeg je 1990. godine prvi razvio Guido van Rossum. U roku od samo 10 godina je prevalio put od investicije nesigurne budućnosti u program kojim se služe vodeće ustanove na području programiranja kao MIT, NASA, IBM, Google, Yahoo itd. On omogućuje programeru više razmišljanja o problemu nego o jeziku. Python je besplatan, open-source je te ima izuzetno dobru potporu, literaturu i dokumentaciju.[3]
Jezične značajke su:

- **Interpretacija međukoda**: Program kompilira kod u niz bytecode-ova koji se spremaju u .pyc datoteke koje su prenosive na bilo koje platforme gdje se mogu izvoditi interpretacijom tog međukoda.
- **Jezik visoke razine**: Ima ugrađene, standarne tipove podataka i tipove podataka visoke razine kao što su liste, n-terci i riječnici.
- **Čista sintaksa**: Sintaksa jezika je jednostavna. Jasno se razlučuje dio po dio koda ponajviše zbog zamjene posebnih znakova za definiranje blokova koda, uvlakama.
- **Napredne značajke jezika**: Objektno je orijentirano programiranje s višestrukim nasljeđivanjem, dohvaćanjem izuzetaka ili iznimki, redefiniranjem standardnih operatora, pretpostavljanjem argumenata, prostora imena, modula i paketa.
- **Proširivost**: Python je pisan u modularnoj C arhitekturi, te se zbog toga može lako proširivati novim značajkama ili API-ima ($eng. Application programming interface$).
- **Bogate knjižice programa**: Uz standardnu instalaciju, Python sadrži preko 200 modula.

4.2 OpenCV

Gary Bradski je izdario OpenCV 1999. godine u Intel-u zbog potrebe ubrazavanja istraživanja i komercijalnih aplikacija računalnog vida u svijetu.
OpenCV (*eng.OpenSourceComputerVisionLibrary*) je open source software library koji je specijaliziran za računalni vid i strojno učenje. OpenCV je izrađen s ciljem omogućavanja jednostavne infrastrukture za aplikacije koje se bave računalnim vidom, te ubrzavanja upotrebe strojne percepcije u svakodnevnim potrebama. Library od OpenCV-a sadrži više od 2500 optimiranih algoritama. Neki od tih algoritama i njihove funkcije su:

- cv.HaarDetectObjects- Ova funkcija traži pravokutna područja u slici koja imaju najveću vjerojatnost pronalaženja objekta koji je predefiniran i vrača ta područja kao sekvencu pravokutnika. Specifičnost ove aplikacije jest da se mora prvo "učiti" uzorak objekta koji je tražen. Objekt se u ovoj funkciji traži kaskadno, tj. prolazi faze traženja od grubog (fokus zauzima veliku površinu i brzo prolazi cijelu sliku) do detaljnog stupnja (fokus zauzima malu površinu, ali sporo prolazi cijelu sliku).

- cv.CalcGlobalOrientation- Ova funkcija izračunava orijentaciu globalnog pomaka određene površine te vraća kut između 0 i 360 stupnjeva. Na početku funkcija izgradi histogram orijentacije i pronalazi osnovnu orijentaciju kao koordinatu maximuma histograma. Nakon toga funkcija izračunava pomak u odnosu na osnovnu orijentaciju kao težinsku sumu od svih vektora. Što je u bližm vremenskom intervalu pomak to je veči utjecaj na globalni pomak.

- cv.CalibrateCamera2- Funkcija koja od više različitih pogleda na istu stvar stvara 3-D prikaz. Kod ove funkcije je bitno da se na slikama nalazi 2D osnovni jednostavni oblici koje će program moći lako prepoznati. Takvi oblici se zovu uzorci za kalibraciju.

- cv.GetCentralMoment- Funkcija koja izračunava težište lika. Posebno je korisna ako je nedefiniran ili izrazito nepravilan oblik objekta koji pratimo.

- cv.DetectEyes- Funkcija koja otkriva ljudske oči na određenoj fotografiji. Jedna od glavnih komponenti za rješavanje problema uklanjanja crvenih očiju. U kombinaciji sa funkcijom cv.Get2D i cv.Treshold ispituje boju očiju te na kraju pomoću funkcije cv.RGB mijenja boju očiju.

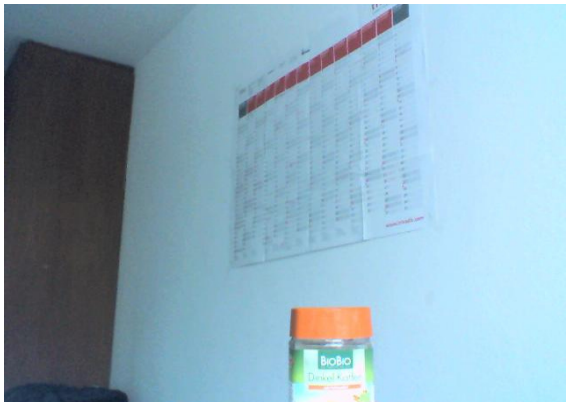
5 Opis algoritma

5.1 Prepoznavanje boje

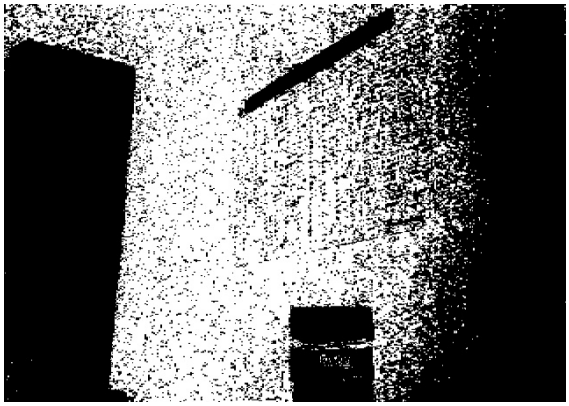
Da bi se mogao napraviti program koji će prepoznat robote po njihovim bojama treba izvesti 4 operacije koje su predefinirane u OpenCV-u.

5.1.1 Tresholding

Osnovni korak kod programiranja algoritma prepoznavanja boja je Tresholding. U doslovnom smislu prevedeno tresholding znači "Određivanje praga". Ovim korakom se odvajaju svi pikseli na slici koji ne zadovoljavaju predodređene uvijete. Najčešće su to ili HSV ili RGB vrijednosti. Pošto postoje tri različita robota koja se moraju pratiti, tj tri različite boje. Ovaj korak se mora ponoviti za svaku boju posebno.Na slikama 3 i 4 nalazi se slika prije i slika poslije Tresholdinga [5]



Slika 3: Slika prije Tresholding-a



Slika 4: Slika poslije Tresholding-a

Na slici 5 se objašnjava kako funkcionira funkcija cv.Tresholding.

- Graf a) prikazuje neobrađene izlazne vrijednosti za pixele. Na y osi se nalaze vrijednosti pixela, dok na x osi se nalaze pixeli.

- Graf b) prikazuje funkciju cv.BinaryTreshold za graf a). Sve vrijednosti iznad zadane vrijednosti dobijaju vrijednost 1 a sve ispod dobijaju vrijednost 0.

  TRESH_BINARY $\quad \text{dst(x,y)}=\begin{cases} \text{maxval} & \text{if src(x,y)} > \text{tresh} \\ 0 & \text{otherwise} \end{cases}$

- Graf c) prikazuje inverz funkcije cv.TresholdBinary

  TRESH_BINARY_INV $\quad \text{dst(x,y)}=\begin{cases} 0 & \text{otherwise} \\ \text{maxval} & \text{if src(x,y)} > \text{tresh} \end{cases}$

- Graf d) prikazuje vrijednosti izlazne funkcije za funkciju cv.Truncate. Ona sve vrijednosti ispod zadane vrijednosti neobrađeno proslijeđuje dalje a vrijednosti iznad zadržava na zadanoj vrijednosti.

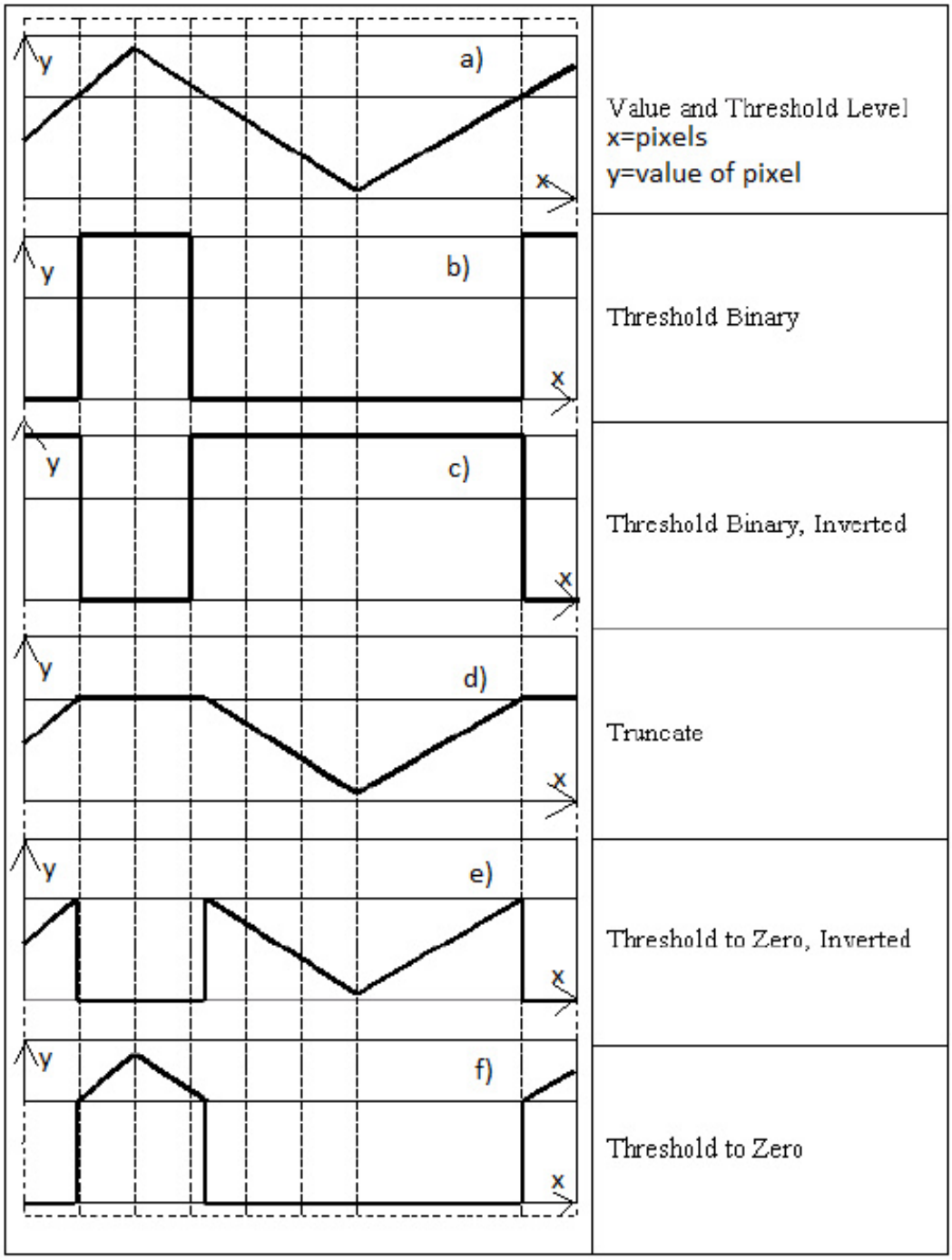
  TRESH_TRUNC $\quad \text{dst(x,y)}=\begin{cases} \text{tresholdl} & \text{if src(x,y)} > \text{tresh} \\ \text{src(x,y)} & \text{otherwise} \end{cases}$

- Graf e) prikazuje ima sličnu funkciju kao cv.Truncate, međutim funkcija cv.TresholdToZero kad ulazne vrijednosti pređu zadanu vrijednost, izlazne vrijednosti padaju na nulu.

  TRESH_TOZERO $\quad \text{dst(x,y)}=\begin{cases} \text{src(x,y)} & \text{if src(x,y)} > \text{tresh} \\ 0 & \text{otherwise} \end{cases}$

- Graf f) prikazuje inverz funkcije cv.TresholdToZero

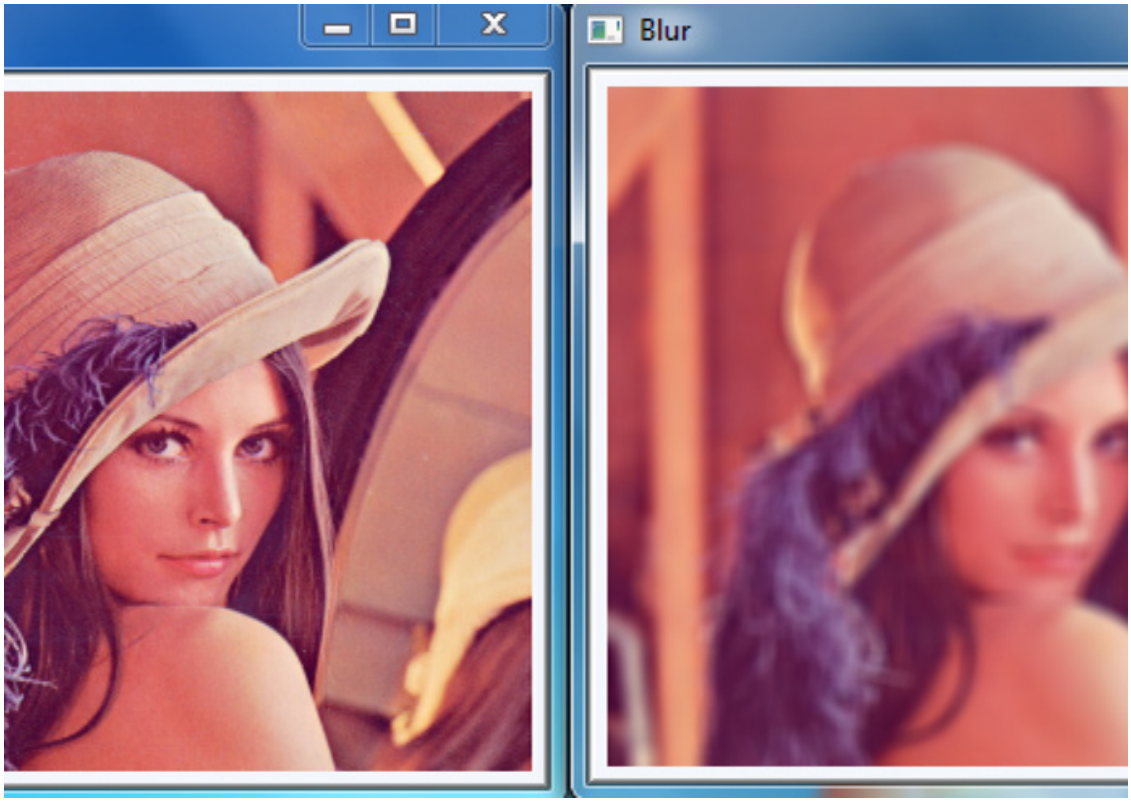
  TRESH_TOZERO_INV $\quad \text{dst(x,y)}=\begin{cases} 0 & \text{otherwise} \\ \text{src(x,y)} & \text{if src(x,y)} > \text{tresh} \end{cases}$



Slika 5: Sve vrste tresholding-a

5.1.2 Smooth

Algoritam Smooth je izrazito važan jer umanjuje, a u isto vrijeme i povečava pogrešku percepcije računalnog vida. Potreban je jer program može analizirati sliku samo pomoču piksela. Pošto kamera ima grešku snimanja onda ovaj korak pomaže u zanemarivanju disperznih piksela, dok u isto vrijeme mijenja granicu željenog objekta kojeg pratimo zbog uglađivanja prijelaza što je pak neželjena posljedica. Na slici 6 nalazi se prikaz djelovanja operacije Smooth. [6]



Slika 6: Slika prije operacije Smooth i poslije

Funkcija Smooth se temelji na Gaussovoj funkciji za računanje (jednadžba (3) za jednu dimenziju i jednadžba (4) za dvije dimenzije) transofrmacije za svaki pixel:

$$G(x) = \frac{1}{\sqrt{2\pi\delta^2}} e^{-\frac{x^2}{2\delta^2}} \tag{3}$$

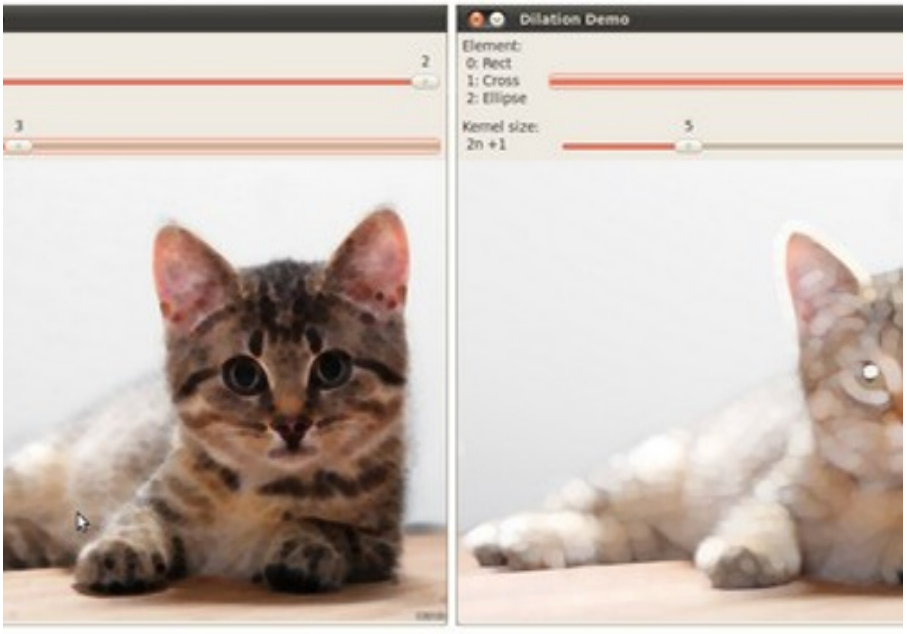
$$G(x) = \frac{1}{2\pi\delta^2} e^{-\frac{x^2+y^2}{2\delta^2}} \tag{4}$$

gdje je x udaljenost od izvorne slike u horizontalnoj osi, y je udaljenost od izvorne slike u vertikalnoh osi, a δ je standardna devijacija Gaussove distribucije.

5.1.3 Eroding and Dilating

Zadnja operacija koja se provodi na slici se uvijek provodi u paru kako bi se riješile sve zaostale nečistoće. Kod Eroding-a se objektu koji promatramo na obrisima Value smanjuje te se podebljavaju obrisi. Dok se kod Dilating-a događa upravo suprotno. Bridovi se stanjuju te se Value povečava. Time obrisi postaju sve svijetliji. Na slici 7 se nalazi neobrađena slika dok se na slici 8 nalaze slike obrađene funkcijama Eroding i Dilating.[7]

Slika 7: Slika prije obrade



Slika 8: Slike poslije obrade: Eroding i Dilating

5.2 Prepoznavanje položaja

Roboti su pravilnog oblika te je zato odabrana funkcija BoundingRect, kako bi se pomoću nje dobilo težište robota na slici. Funkcija cv.BoundingRect od obrađene slike odabire optimalni pravokutnik koji obuhvaća cijelu konturu robota crvene boje.
U ovom koraku funkcija od skupine pixela koje je funkcija cv.FindContours prepoznala kao pixele crvene boje se analizira kako bi došla do najoptimalnijeg pravokutnika.

```
bound\_rect = cv.BoundingRect(list(contour))
```

U ovom koraku se određuju sva četri vrha optimalnog pravokutnika:

```
pt1 = (bound\_rect[0], bound\_rect[1])
pt2 = (bound\_rect[0] + bound\_rect[2], bound\_rect[1] + bound\_rect[3])
```

Korak u kojem se računa težište robota:

```
centroidx=cv.Round((pt1[0]+pt2[0])/2)
centroidy=cv.Round((pt1[1]+pt2[1])/2)
```

Naknadno se if petljom razdijeljuju na pripadajuću boju, tj.povezuju se robot, boje i položaj. Na slici 9 prikazan je primjer prepoznavanja crvene boje na predmetu. [8]

Slika 9: Primjer određivanja težista

5.3 Slanje informacija

Za problem slanja informacija putem serijske veze odabran je algoritam za RS232 veze. U početku koda potrebno je definirati brzinu dijeljenja podataka, veličinu paketa u kojima će se dijeliti podaci te port na koj će se poslati podaci.[9]

```
ser = serial.Serial(
port='/dev/ttyUSB1'
baudrate=9600
parity=serial.PARITY\_ODD
stopbits=serial.STOPBITS\_TWO
bytesize=serial.SEVENBITS)
```

Navedeni kod se kasnije implementira u programu:

```
ser.open()
ser.write(red,blue,yellow)
ser.close()
```

Na ovakav na način se otvara veza gdje se šalju podaci nakon čega se veza zatvara.

5.4 Orijentacija

Kod orijentacije se treba uzeti u obzir da su roboti koje treba prepoznati kompliciranog oblika kao što se vidi na slici 2. Konkavnost s obje strane nije dovoljno različita da se može prepoznati zbog pogreške kamere. Ideja koja je uzeta za rješenje je korištenje funkcije cv2.minAreaRect() s kojom se konture mogu obuhvatiti sa najboljim pravokutnikom. Prednost ovog oblika pravokutnika je da se rotira ovisno o najoptimalnijem slučaju. [10]

6 Problematika Open Source programa

Jedan od glavnih problema kod programiranja u OpenCV-u je problematičan odnos programiranja posebnih datoteka za programski razvoj Python-a. Zbog daljnjeg razvoja Python-a, OpenCV često ne izvršava svoje funkcije u svakoj verzij stoga je ptorebno imati odgovarajuču verziju OpenCV-a koja je u potpunosti kompaktibilna sa Pythonom. Zbog takviih problema priloženi kod nije ispunio funkciju orijentacije. U ovom slučaju je odabrana funkcija cv2.minAreaRect(). Primjer kako bi se trebala koristiti funkcija cv2.minAreaRect je:

```
1 rect = cv2.minAreaRect(cnt)
2 box = cv2.cv.BoxPoints(rect)
3 box = np.int0(box)
4 cv2.drawContours(im,[box],0,(0,0,255),2)
```

Varijabla cnt u ovom slučaju predstavlja skup točaka koji su dobiveni funkcijom cv2.FindContours. Ona bi trebala pratiti rotaciju robota, a usmjerenje bi se izračunalo udaljenošću između težišta i stranica robota s čime bi se jednostavno dobila orijentacija. No zbog gore navedenih razloga ovaj algoritam je bio jedan od algoritama koji nisu izvrĹ˘avali svoju funkciju.

7 Zaključak

Prepoznavanje različitih boja je vrlo važan korak u bilo kojem sustavu koji se koristi u traženju i raspoznavanju objekata, posebice jer pomaže u isticanju značajnih karakteristika traženih objekata. Zbog toga dobro prepoznavanje slike može biti od iznimnog značaja za poboljšanje performansi drugih algoritama koji se bave analizom slike. Međutim važno je da postupak prepoznavanja ne preopterečuje cijeli sustavs obzirom na njegove performanse, posebice ako postoji potreba za obradom velike količine informacija ili ako je riječ o sustavu koji radi u realnom vremenu. Zato je vrlo važno kvalitetno izraditi sustav za segmentaciju slike po pitanju kvalitete rezultata, i po pitanju brzine izvođenja. Implementacija koja je izrađena u sklopu ovog završnog rada temelji se na vezi između OpenCV-a i Python-a koji se pokazao vrlo nestabilnim ako verzije programa nisu usklađene. Odabir boje kao jedinog kriterija za segmentaciju pokazao se kao dosta problematičnim, te je bilo potrebno primjenjivati više korekcija i nadogradnji na rezultate dobivene takvom segmentacijom kako bi oni bili primjenjivi kao kriterij za raspoznavanje boja. Velika razlika crvene, plave i žute u HSV sustavu boja olakšalo je raspoznavanje.

8 Sažetak

8.1 Program za prepoznavanje e-MIR mobilnih robota kamerom

Program je napravljen u python-u te je uz to instaliran dodatni library OpenCV. Preko kamere sepreuzimaju slike koje se prvo prerađuju te nakon toga analiziraju. Iz toga dobiveni podaci suže za otkrivanje lokacijje, usmjerenja i boje robota. Preko rs232 porta se onda informacije šalju drugom računalu.

8.2 Ključne riječi

OpenCV, računalni vid, segment, RGB prostor boja,HSV prostor boja, detekcija,e-MIR roboti.

Literatura

[1] M.Crneković,D.Zorc,Z.Kunica: Research of mobile robot behavior with eMIR, University of Zagreb, Zagreb ,Croatia.

[2] Erik G. Learned-Miller: Introduction to Computer Visiony University of Massachusetts, January19,2011.

[3] Python, http://zrno.fsb.hr/katedra/download/materijali/623.pdf, 10.09.2013.

[4] OpenCV, http://opencv.org/about.html, 05.09.2013.

[5] Basic Thresholding Operations, http://docs.opencv.org/doc/tutorials /imgproc/threshold/threshold.html, 01.02.0213.

[6] Smoothing Images, http://docs.opencv.org/doc/tutorials/imgproc /gausian_median_blur_bilateral_filter/gausian_median_blur_bilateral_filter.html, 25.08.2013.

[7] Eroding and Dilating, http://docs.opencv.org/doc/tutorials/imgproc/erosion _dilatation/erosion_dilatation.html, 26.08.2019.

[8] Creating Bounding boxes and circles for contours, http://docs.opencv.org/doc /tutorials/imgproc/shapedescriptors/bounding_rects_circles/ bounding_rects_circles.html, 02.09.2013.

[9] Serial RS232 connections in Python, http://www.varesano.net/blog/fabio/serial%20rs232%20connections%20python 15.08.2013.

[10] Contours-2:Brotherhood, http://opencvpython.blogspot.com/2012/06 /contours-2-brotherhood.html, 03.09.2013.

```
from cv2 import cv
import cv2
import time
import serial
# configure the serial connections (the parameters differs on the device you are
connecting to)
#ser = serial.Serial(
#   port='/dev/ttyUSB1',
#   baudrate=9600,
#   parity=serial.PARITY_ODD,
#   stopbits=serial.STOPBITS_TWO,
#   bytesize=serial.SEVENBITS
#)
posx=0
posy=0
def getthresholdedimg(im):
    global imghsv
    imghsv=cv.CreateImage(cv.GetSize(im),8,3)
    cv.CvtColor(im,imghsv,cv.CV_BGR2HSV)
    imgyellow=cv.CreateImage(cv.GetSize(im),8,1)
    imgblue=cv.CreateImage(cv.GetSize(im),8,1)
    imgred=cv.CreateImage(cv.GetSize(im),8,1)
    imgthreshold=cv.CreateImage(cv.GetSize(im),8,1)
    cv.InRangeS(imghsv,cv.Scalar(160,100,100),cv.Scalar(180,255,255),imgred)
    cv.InRangeS(imghsv,cv.Scalar(10,100,100),cv.Scalar(50,255,255),imgyellow)
    cv.InRangeS(imghsv,cv.Scalar(90,100,100),cv.Scalar(140,255,255),imgblue)
    cv.Add(imgyellow,imgblue,imgthreshold)
    cv.Add(imgthreshold,imgred,imgthreshold)
    return imgthreshold
capture=cv.CaptureFromCAM(0)
frame = cv.QueryFrame(capture)
frame_size = cv.GetSize(frame)
test=cv.CreateImage(cv.GetSize(frame),8,3)
cv.NamedWindow("Real",0)
blue=[]
yellow=[]
red=[]
while(1):
    color_image = cv.QueryFrame(capture)
    imdraw=cv.CreateImage(cv.GetSize(frame),8,3)
    cv.SetZero(imdraw)
    cv.Flip(color_image,color_image,1)
    cv.Smooth(color_image, color_image, cv.CV_GAUSSIAN, 3, 0)
    imgyellowthresh=getthresholdedimg(color_image)
    cv.Erode(imgyellowthresh,imgyellowthresh,None,3)
    cv.Dilate(imgyellowthresh,imgyellowthresh,None,10)
    storage = cv.CreateMemStorage(0)
    contour = cv.FindContours(imgyellowthresh, storage, cv.CV_RETR_CCOMP,
cv.CV_CHAIN_APPROX_SIMPLE)
    points=[]
#   cv.DrawContours(color_image,contour, cv.RGB(255,0,0), cv.RGB(255,0,0),2)
    while contour:
        bound_rect = cv.BoundingRect(list(contour))
        contour = contour.h_next()
        pt1 = (bound_rect[0], bound_rect[1])
        pt2 = (bound_rect[0] + bound_rect[2], bound_rect[1] + bound_rect[3])
        points.append(pt1)
        points.append(pt2)
        cv.Rectangle(color_image, pt1, pt2, cv.CV_RGB(255,0,0), 2)
        centroidx=cv.Round((pt1[0]+pt2[0])/2)
```

```
        centroidy=cv.Round((pt1[1]+pt2[1])/2)
        if (10<cv.Get2D(imghsv,centroidy,centroidx)[0]<50):
            yellow.append((centroidx,centroidy))
        elif (90<cv.Get2D(imghsv,centroidy,centroidx)[0]<140):
            blue.append((centroidx,centroidy))
        elif (160<cv.Get2D(imghsv,centroidy,centroidx)[0]<180):
            red.append((centroidx,centroidy))
        time.sleep(0.1)
#        ser.open()
#        ser.write(red,blue,yellow)
#        ser.close()
        print red,blue,yellow
        red=[]
        blue=[]
        yellow=[]
    cv.Add(test,imdraw,test)
    cv.ShowImage("Real",color_image)
    if cv.WaitKey(15)%0x100==27:
        cv.DestroyWindow("Real")
        break
```