

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 686

**PREPOZNAVANJE STRUKTURE KAVEZA
ZA UZGOJ RIBA U VIDEO SLICI**

Roko Romić

Zagreb, lipanj 2013.

*Zahvaljujem mentoru prof. dr. sc. Zoranu Vukiću, na suradnji,
stručnoj pomoći i savjetima tijekom izrade diplomskog rada.
Također se zahvaljujem doc. dr. sc. Nikoli Miškoviću na suradnji i
savjetima tijekom izrade diplomskog rada.*

Sadržaj

Uvod.....	1
1. OpenCV	2
1.1 Povijest OpenCV-a.....	2
1.2 Instalacija OpenCV	3
2. Korištene tehnike kod obrade slike	9
2.1. Obrada slike	9
2.1.1. Pregled ostvarenog sustava	11
3. Programska implementacija.....	29
3.1. Implementacija za obradu slike.....	29
3.2. Implementacija za obradu video slike	32
4. Eksperimentalni rezultati.....	34
4.1. Primjeri uspješne detekcije	34
4.2. Primjeri djelomično uspješne detekcije	37
4.3. Lažne detekcije.....	38
5. Stereo vizija	40
5.1. Triangulacija	40
5.2. Epipolarna geometrija.....	42
5.3. Esencijalna i fundamentalna matrica	43
Zaključak	45
Literatura	46
Sažetak	47
Summary	48
Skraćenice	49
Privitak	50

Uvod

Razvojem računarstva i tehnike, danas ih je moguće iskoristiti za rješavanje problema koji su se nekoć činili gotovo nerješivim. Kao primjer takvih problema je i obrada te detekcija objekta u slici ili videu, to jest *raspoznavanje objekta*. Raspoznavanje uzoraka je znanstvena disciplina u području računarske znanosti i umjetne inteligencije čiji je cilj klasifikacija objekata u kategorije ili razrede. Postoji velik broj različitih algoritama za klasifikaciju objekata. Podvodna robotika je disciplina koja ima sve veću ulogu u istraživanju podvodnog svijeta, koji danas nije u potpunosti istražen, već skriva još mnogo tajni.

Primjena ROV-a je sve opsežnija, korištenje u marikulturi¹ je danas izazovno područje istraživanja. Kako se bespilotne ronilice kreću u okolini s poznatom strukturuom, u ovom slučaju radi se o kavezu za uzgoj riba, u većini slučajeva riječ je od četvrtastim mrežnim rešetkama, potrebno je iskoristit tu informaciju kako bi se povećala kvaliteta navigacijskih algoritama.

Cilj ovog diplomskog rada je implementirati prikladan algoritam za obradu video slike, kojim će se moći detektirati pravilne strukture kaveza za uzgoj riba. Za razvoj algoritma koristit će se programsko okruženje OpenCV, te Microsoft Visual Studio 2010. Uz samu detekciju pravilne strukture kaveza za uzgoj ribe, potrebno je i istražiti i mogućnost određivanja udaljenosti bespilotne ronilice/kamere od kaveza na temelju video slike.

U prvom dijelu biti će opisano programsko okruženje OpenCV, dok će se drugo poglavlje sastojati od programske implementacije. Te će rad završiti analizom eksperimentalno dobivenih rezultata te istražiti mogućnost određivanja dubine slike.

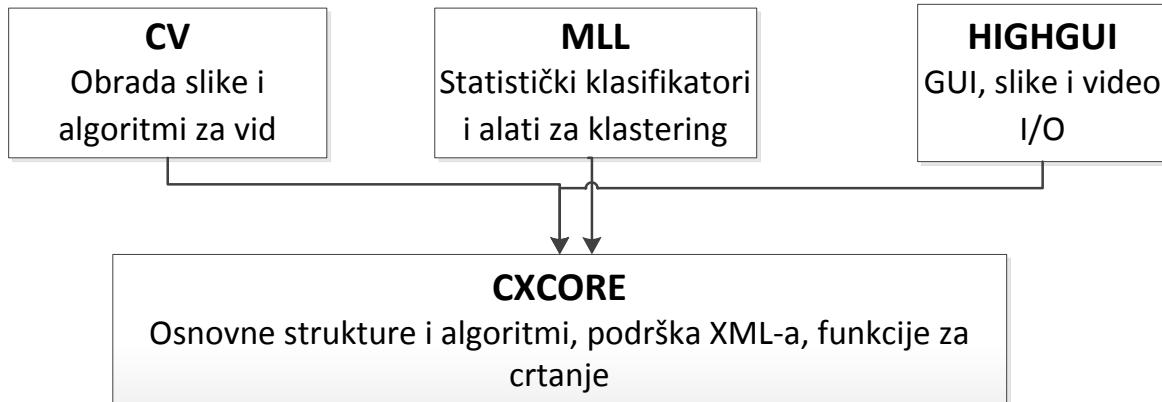
1. OpenCV

1.1 Povijest OpenCV-a

OpenCV (eng. Open Computer Vision library) je biblioteka (eng. libarary) otvorenog koda (eng. open source) namijenjena za računalni vid. Službeno je predstavljena 1999. godine od strane Intel-a. Biblioteka je realizirana u C i C++ programskim jezicima. Ujedno je i neovisna o izboru operacijskog sustava, to jest moguće ju je koristiti na Windows, Linux, Mac Os X, Android te iOS operacijskim sustavima. Razvijanje korisničkih aplikacija je moguće u C, C++, Pythonu, Ruby, Java, Matlab programskim jezicima. Glavni cilj programera pri stvaranju biblioteke bio je omogućiti korisnicima jednostavan i brz razvoj aplikacija na području računalnog vida. Jedno od bitnih svojstava aplikacija izgrađenih na ovoj biblioteci je mogućnost korištenja višejezgrenih procesora. Biblioteka sadrži preko 500 funkcija koje obuhvaćaju skoro sva područja današnje primjene računalnog vida. Neke primjene su: prepoznavanje lica, mobilna robotika, prepoznavanje gesta, identifikacija objekta, prepoznavanje, stereo vizija, praćenje objekta...

Neki od glavnih modula su:

- cxcore – funkcionalnost jezgre,
- cv – procesiranje slika i računalni vid,
- highgui – korisničko sučelje te čitanje i zapis slika,
- ml – modul za strojno učenje.

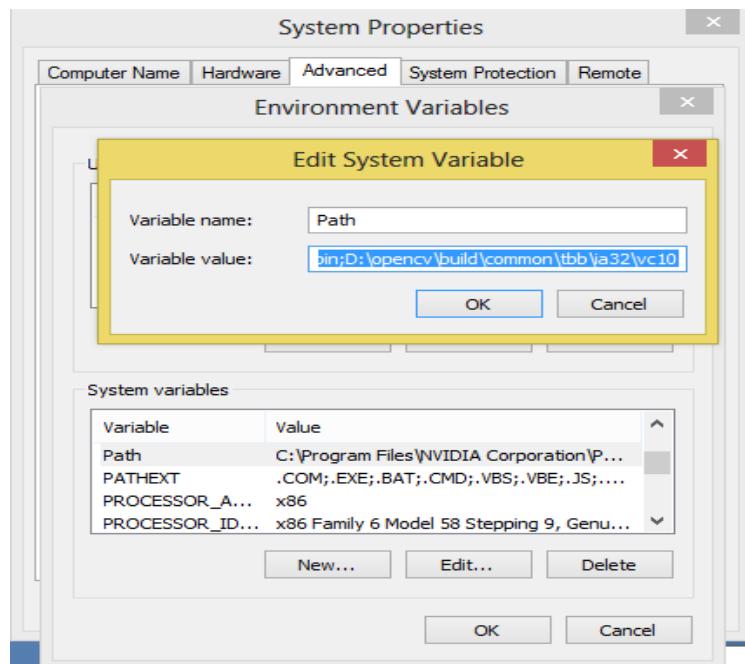


Slika 1. Osnovna struktura OpenCv-a.

1.2. Instalacija OpenCV

Koristio se programski paket Microsoft Visual Studio 2010, te OpenCV biblioteka OpenCV 2.4.2 [1]. Operacijski sustav je Windows 8 32-bit.

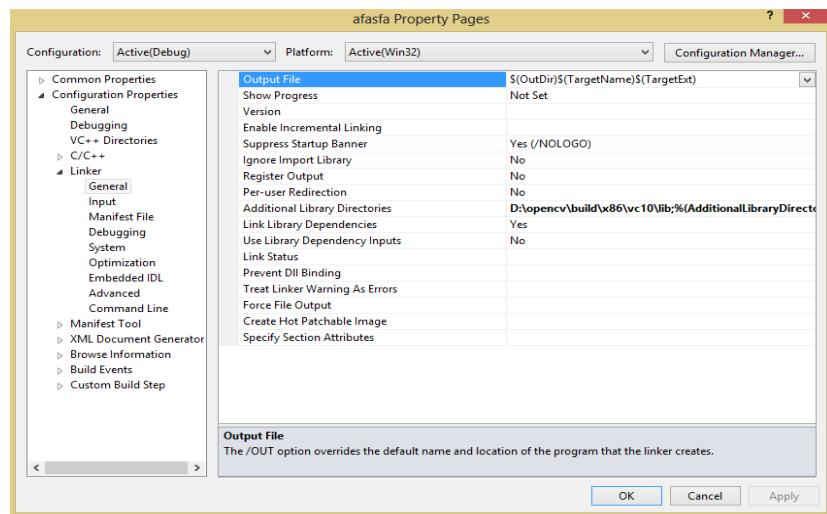
U prvom koraku potrebno je postaviti sistemske varijable, do kojih se dođe *Control Panel -> System and Security -> System ->Advanced System settings*, pa onda kartica *Advanced -> System Variables PATH* dodati slijedeće putanje: „;D:\opencv\build;D:\opencv\build\x86\vc10\bin;D:\opencv\build\common\tbb\ia32\vc10 „, pri čemu je D:\opencv lokacija gdje se nalazi biblioteka OpenCV-a.



Slika 2. Prikaz postavljanja sistemske varijable

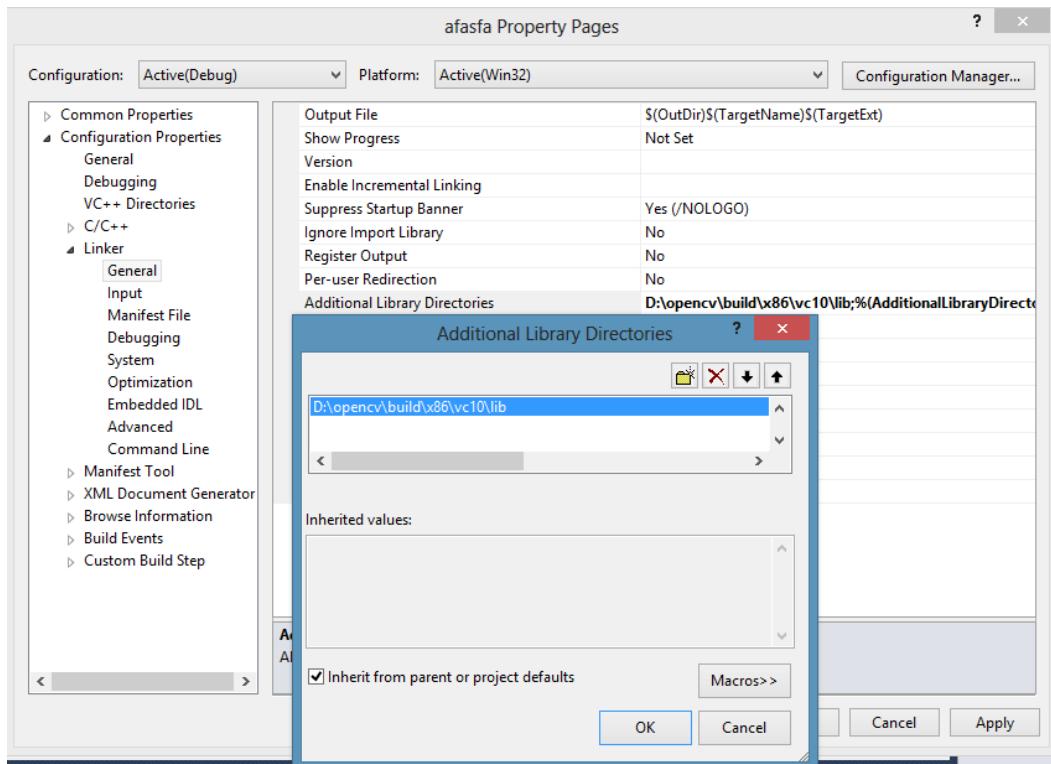
Drugi korak kod instalacije je pokretanje Microsoft Visual Studio-a te kreiranje novog projekta. Odabir *Visual C++ -> Win32 Console Application*.

U trećem koraku je potrebno postaviti OpenCV biblioteke u novi projekt stvoren u MS Visual Studio-u. Da bi se to postiglo potrebno je podesiti *linker* u MS Visual Studio-u. To se postiže na slijedeći način. U kartici (eng. Tab) Project odabere se *Properties*.



Slika 3. Prikaz Properties kartice

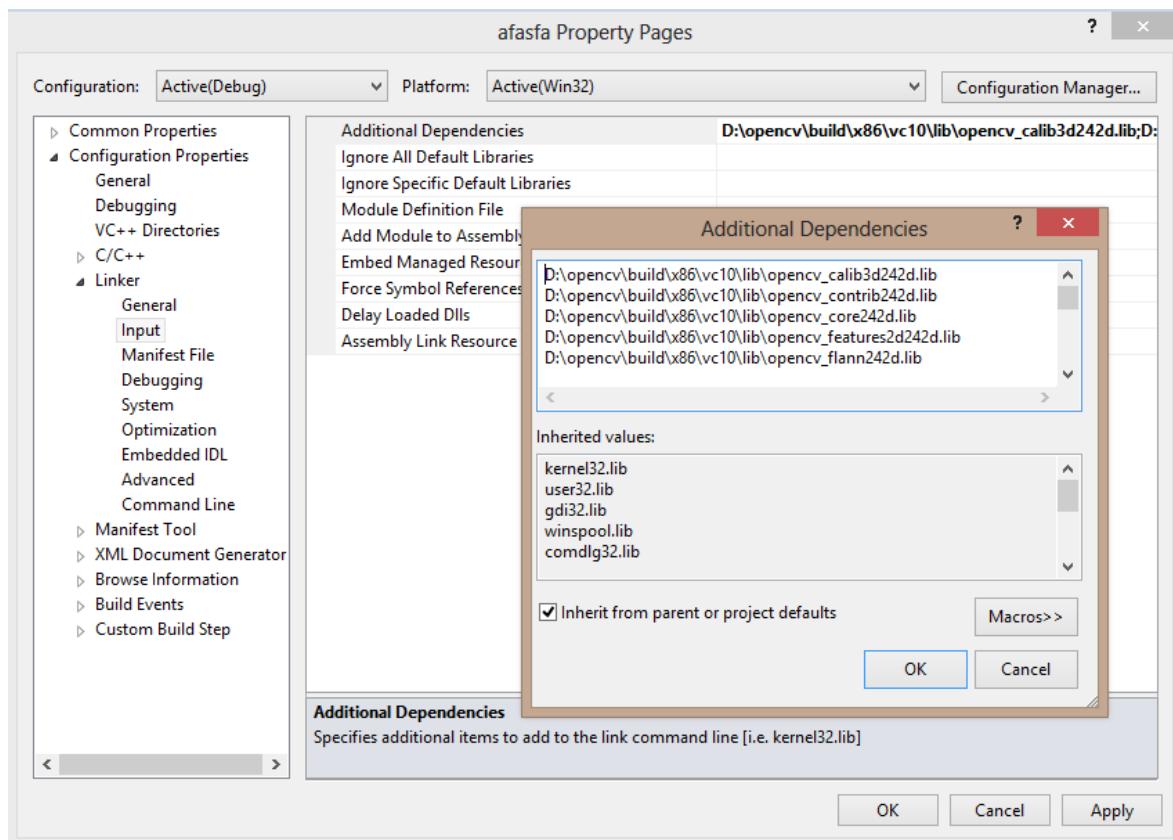
Dalje je potrebno kod *Linker* -> *General* -> *Additional Library Directories* postaviti slijedeću putanju: „D:\opencv\build\x86\vc10\lib“.



Slika 4. Postavljanje dodatnih biblioteka

U slijedećem koraku potrebno je postaviti u *Linker -> Input -> Additional Dependencies* slijedeće biblioteke:

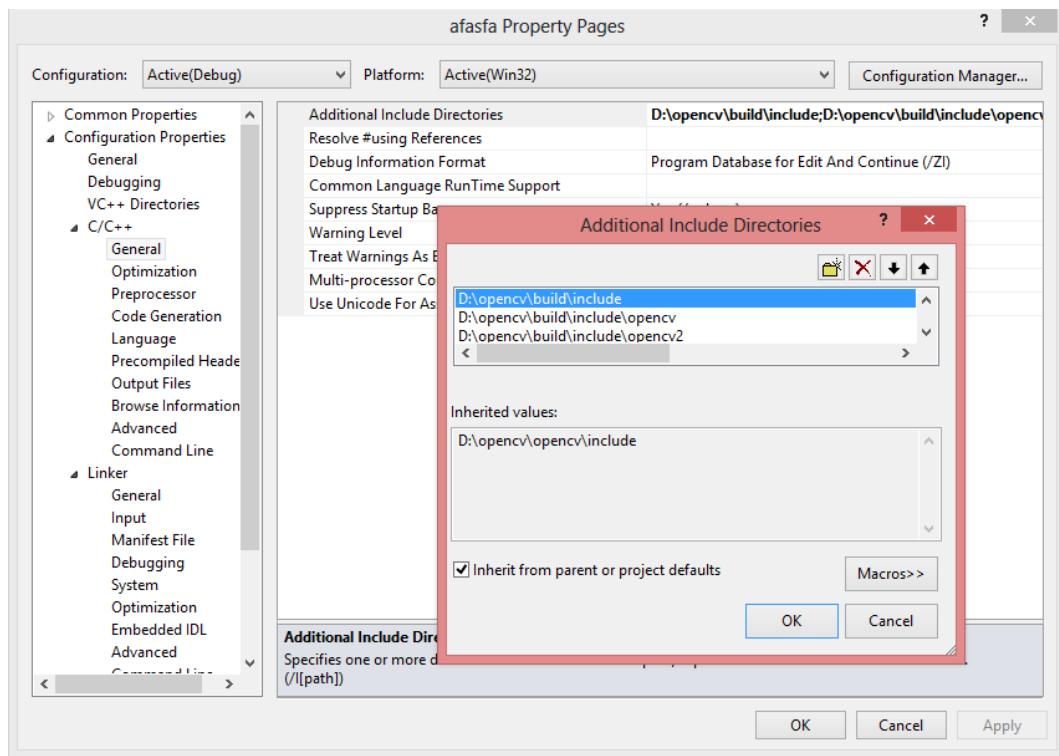
```
D:\opencv\build\x86\vc10\lib\opencv_calib3d242d.lib  
D:\opencv\build\x86\vc10\lib\opencv_contrib242d.lib  
D:\opencv\build\x86\vc10\lib\opencv_core242d.lib  
D:\opencv\build\x86\vc10\lib\opencv_features2d242d.lib  
D:\opencv\build\x86\vc10\lib\opencv_flann242d.lib  
D:\opencv\build\x86\vc10\lib\opencv_gpu242.lib  
D:\opencv\build\x86\vc10\lib\opencv_haartraining_engined.lib  
D:\opencv\build\x86\vc10\lib\opencv_highgui242d.lib  
D:\opencv\build\x86\vc10\lib\opencv_imgproc242d.lib  
D:\opencv\build\x86\vc10\lib\opencv_legacy242d.lib  
D:\opencv\build\x86\vc10\lib\opencv_ml242d.lib  
D:\opencv\build\x86\vc10\lib\opencv_nonfree242d.lib  
D:\opencv\build\x86\vc10\lib\opencv_objdetect242d.lib  
D:\opencv\build\x86\vc10\lib\opencv_photo242d.lib  
D:\opencv\build\x86\vc10\lib\opencv_stitching242d.lib  
D:\opencv\build\x86\vc10\lib\opencv_ts242d.lib  
D:\opencv\build\x86\vc10\lib\opencv_video242d.lib  
D:\opencv\build\x86\vc10\lib\opencv_videostab242d.lib
```



Slika 5. Postavljanje dodatnih zavisnih biblioteka potrebnih za rad

Te zadnji korak instalacije. Potrebno je dodati OpenCV *include direktorije* u novi projekt stvoren u MS Visual Studio-u. Odabere se kartica C/C++ u *Property Pages*-u te se pod novu karticu *General -> Additional Include Directories* dodaju slijedeće putanje:

```
D:\opencv\build\include  
D:\opencv\build\include\opencv  
D:\opencv\build\include\opencv2
```



Slika 6. Postavljanje dodatnih direktorija

Sada je MS Visual Studio povezan s OpenCV-om, te se može početi sa pisanjem algoritma, u ovom slučaju pisati će se u C programskom jeziku.

2. Korištene tehnike kod obrade slike

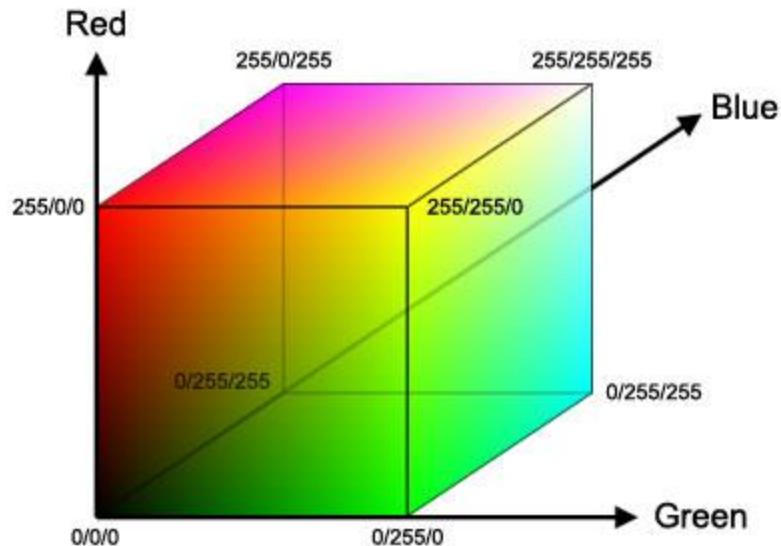
Zbog jednostavnosti te mogućnosti isprobavanja različitih solucija vezanih uz detekciju kaveza, krenulo se od obrade slike, to jest detekcije kaveza na danoj slici. Kao ulazne slike koristile su se slike dostupne na internetu. Nakon što se odabrala najbolje rješenje za detekciju kaveza kod obrade slike, to isto rješenje se doradilo za obradu video slike. Što je ujedno i cilj ovog diplomskog rada. Također kao ulazni video materijali koristili su se oni dostupni na video servisu kao što je *Youtube*.

Problema kod detekcije ima svakakvih, kao što je loše osvjetljenje, koje je najčešće uzrok lažne detekcije ili nemogućnosti detekcije. Neki drugi problemi detekcije nastaju zbog loših vremenskih uvjeta, kretanja drugih objekata blizu objekta koji je cilj detekcije, te postoje još mnogi drugi problemi. Svi ti problemi se mogu ukloni, najčešće ne u potpunosti, korištenjem različitih metoda, kao što je uklanjanje šuma, izjednačavanje histograma, eliminacija položaja, korištenjem različitih filtara itd. U ovom radu pobliže će se spomenuti neke metode.

2.1. Obrada slike

Kvalitetno predprocesirana slika na ulazu sustava uvelike ovisi o konačnim rezultatima samog sustava, stoga ćemo se u ovome poglavlju posvetiti postupcima predprocesiranja tj. obrade ulazne slike. Ostvareni sustav fleksibilan je što se tiče formata ulaznih slika, tako su podržani danas najčešće korišteni formati slika: BMP, DIB, JPEG, JPG, JPE, PNG, PBM, PGM, PPM, SR, RAS, TIFF i TIF. Sustav podržava rad sa ulaznim slikama različitih dimenzija. Ulazne slike u sustav učitavaju se u 24bitnom RGB formatu, pri čemu svaki od kanala sadrži osam bita, tj. vrijednosti od 0 do 255. 24bitni RGB format možemo promatrati kao kocku prikazanu na slici 6. Kocka je smještena u cjelobrojni trodimenzionalni sustav koordinata R (eng. red, hrv. crvena), G (eng. green, hrv. zelena) i B (eng. blue, hrv. plava). Bridovi kocke su duljine 255. Boju pojedinog piksela slike određuju njegove pripadne cjelobrojne koordinate u trodimenzionalnom RGB sustavu. Na slici 7. se vidi vidjeti 24bitni trodimenzionalni RGB sustav sa primjerom dodjele boje

jednom pikselu. Rubne vrijednosti boja 24bitnog RGB sustava sa pripadnim koordinatama dane su u tablici 1.

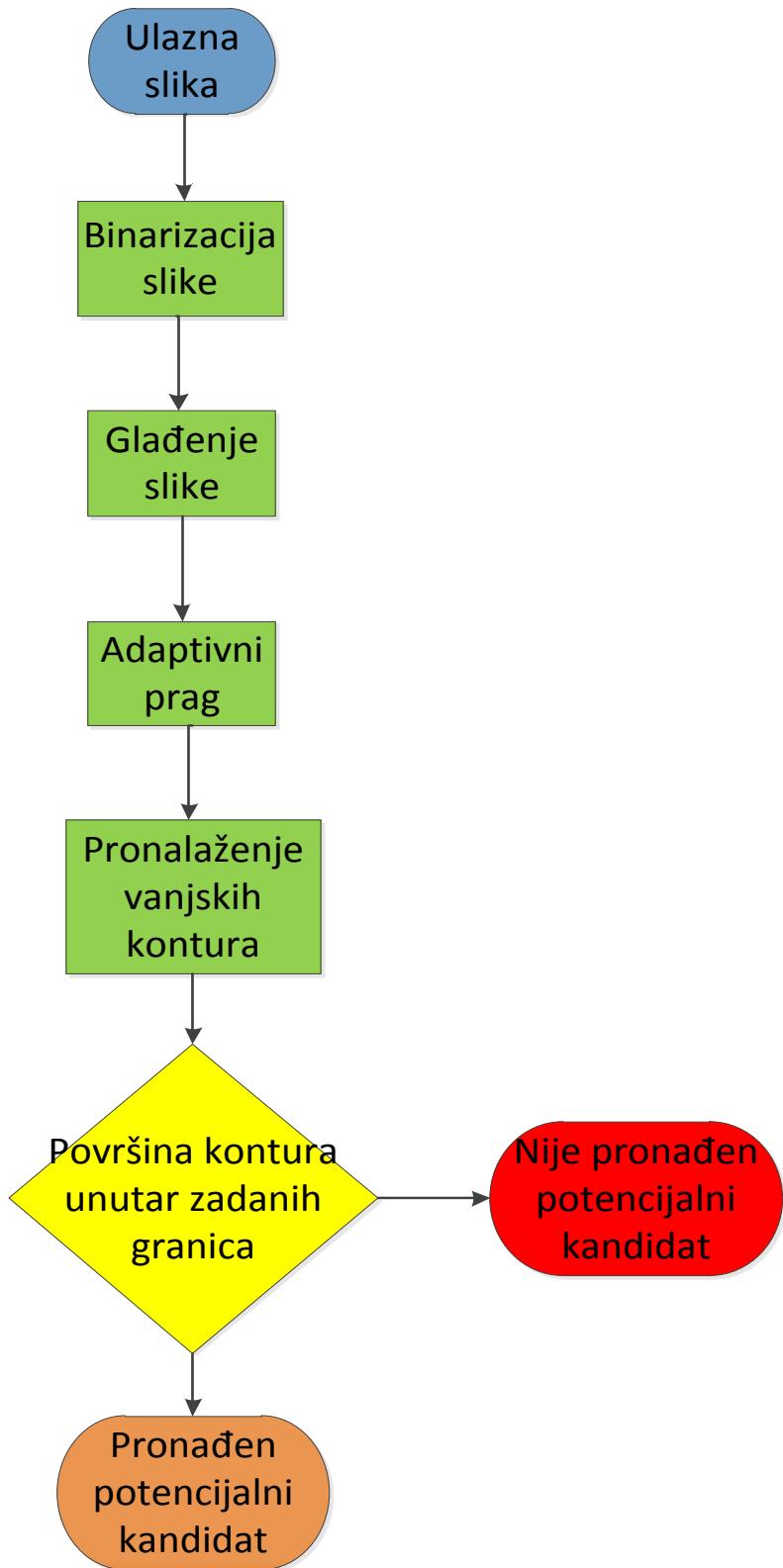


Slika 7. 24bitni RGB sustav sa primjerom dodjele boje pikselu [2]

Tablica 1. Rubne vrijednosti boja 24bitnog RGB sustava

Koordinate (R,G,B)	Boja
(0, 0, 0)	crna
(255,255,255)	bijela
(255,0,0)	crvena
(0,255,0)	zelena
(0,0,255)	plava
(255,255,0)	žuta
(0,255,255)	cijan
(255,0,255)	ružičasta

2.1.1. Pregled ostvarenog sustava



Slika 8. Shema ostvarenog sustava

Na slici 8. je prikazana shema ostvarenog sustava. Realiziran je na način da se sastoji od 7 segmenata. Segmenti su redom: *Ulazna slika*, *Binarizacija slike*, *Glađenje slike*, *Adaptivni prag*, *Pronalaženje vanjskih kontura*, *Površina kontura*, te na kraju *Potencijalni kandidat* ili jednostavno nije pronađen isti. Segmenti ostvarenog sustava *Glađenje slike* i *Adaptivni prag* spadaju u dio za filtriranje, to jest uklanjanje smetnji u slici. U dalnjem dijelu rada bit će objašnjen svaki od segmenata sustava.

2.1.1.1 Binarizacija slike

Proces smanjenja broja boja na dvije boje (a time i na jedan kanal) naziva se binarizacija. Na ovaj proces se može gledati kao na prvi korak segmentacije jer se važniji slikovni elementi koji tvore objekt korišten u kasnijim fazama razdvajaju od pozadine. Ideja je vrlo jednostavna – tamnije dijelove slike obojiti jednom bojom (obično crnom), a svjetlijе dijelove drugom bojom (bijelom). Pritom je potrebno obojiti cijelu površnu slike. Postavlja se pitanje kako odrediti koji dijelovi su tamniji (objekt), a koji svjetlijи te u odnosu na što su oni tamniji odnosno svjetlijи (pozadina). Uvodi se pojam vrijednost praga. Vrijednost praga predstavlja referentnu vrijednost intenziteta boje. Ukupni intenzitet elementa računa se kao suma intenziteta svih triju boja skaliranih određenim udjelom. Standardni udjeli (k_r , k_g i k_b) su 30% crvene boje, 59% zelene boje i 11% plave boje. Svaki element tamniji od praga svrstava se pod objekt, dok se svaki svjetlijи smatra pozadinskim.

Algoritam 1. Binarizacija slike

1. Postavi vrijednost praga threshold = CONST.
2. Postavi udjele boja k_r , k_g i k_b na definirane vrijednosti.
3. Za svaki element slike napravi sljedeće:
 - (a) Neka je grey = $\text{red}_{i,j} * k_r + \text{green}_{i,j} * k_g + \text{blue}_{i,j} * k_b$
 - (b) Ako je $\text{grey} \leq \text{threshold}$
 - onda pixel_{i,j} stavi u skup objekt
 - inače pixel_{i,j} stavi u skup pozadina
4. Stvori binariziranu sliku bojeći elemente s indeksima jednakim indeksima elemenata u skupu objekt crnom bojom, a elemente s indeksima jednakim indeksima elemenata u skupu pozadina bijelom bojom.

Algoritam 1. opisuje proces binarizacije slike u boji. Svaki element se postavi na jednu od dvije boje ovisno o vrijednosti praga. Odmah se može uočiti nedostatak algoritma – ograničenost predefiniranom nepromjenjivom vrijednosti praga. Uspješnost binarizacije ovisi o vrijednosti praga i osvjetljenju slike.



Slika 9. Primjer binarne slike

2.1.1.2 Glađenje slike

Drugi korak u izvedbi algoritma je proces glađenja slike. Glađenje je postupak filtriranja slike koji na izlazu daje blago zamagljeniju nego što je to bila početna slika. Glađenjem slike se eliminiraju lažni rubovi na slici što je bitno za kasniju detekciju rubova. Lažne rubove je moguće eliminirati pomoću linearog niskopropusnog filtra kakav je Gaussov filter. Gaussov filter je u izvornom obliku dvodimenzionalan i koristi Gaussovu funkciju normalne raspodjele kako bi njome popunio matricu koju će primijeniti na piksele slike. Konačna vrijednost svakog piksela se određuje operacijom konvolucije na način da svaki element dobivene matrice množi odgovarajući piksel u susjedstvu, a zbroj svih tih umnožaka je jednak novoj vrijednosti promatranog piksela. Konvolucijom je svaki piksel određen svojim susjedstvom. Susjedni pikseli koji su bliže promatranom pikselu će imati veći utjecaj na konačnu vrijednost promatranog piksela od onih koji su dalje zbog djelovanja normalne raspodjele. Isti efekt koji se postiže primjenom dvodimenzionalnog Gaussovog filtra je moguće postići sa dva jednodimenzionalna filtra koji se popunjavaju vrijednostima Gaussove jednodimenzionalne funkcije (1). Dvodimenzionalna Gaussova funkcija je jednaka produktu dviju jednodimenzionalnih funkcija (1). Kod ovakvog pristupa slika se prvo sa jednodimenzionalnim vektorom prođe u jednom smjeru, a zatim se na tako izmijenjenu sliku primjeni isti vektor u drugom smjeru. Jednodimenzionalna Gaussova funkcija određuje vrijednosti svakog piksela na temelju njegovog susjedstva i time je intenzitet svakog piksela usklađen s okolinom. Analitički, jednodimenzionalna Gaussova funkcija može se izraziti sljedećim izrazom:

$$G(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} \quad (1)$$

U izrazu (1), parametar σ je standardna devijacija Gaussove funkcije, x je udaljenost od centralnog piksela po x osi ili po y osi ovisno da li se vrši horizontalno ili vertikalno glađenje. Duljina vektora jednodimenzionalne Gaussove funkcije treba biti primjerena jer utječe na kvalitetu i brzinu izvođenja glađenja slike i neparna zbog

jednoznačnog određivanja centralnog piksela. Gaussova razdioba je definirana na beskonačnom intervalu, ali je taj interval kod realne implementacije potrebno ograničiti. Zato se koristi preciznost od 3σ (3 sigma) koja osigurava točnost Gaussove razdiobe od 99.73%. Uzimajući u obzir oba intervala duljine 3 sigma i pazeći da duljina vektora mora biti neparna. Poželjno je da se vrijednosti unutar vektora podese na način da njihov zbroj iznosi jedan jer u protivnom originalna slika gubi na svjetlini zbog gubljenja dijela vrijednosti koeficijenata. Da bi ukupan zbroj koeficijenata iznosio jedan svaki element vektora se dijeli s ukupnim zbrojem. Glaćenje se vrši konvolucijom vektora sa slikom prolaskom vektora duž cijele slike vodoravno redak po redak počevši od donjeg lijevog kuta pa sve do gornjeg desnog dva puta, jednom vodoravno okrenutim vektorom, a zatim okomito okrenutim vektorom koristeći vodoravno zaglađenu sliku kao ulaz. Gaussov vektor veće duljine smanjuje brzinu izvođenja algoritma te uzrokuje veću zamagljenost izlazne slike, povećavajući tako grešku lokalizacije, ali i smanjujući utjecaj štetnog šuma.

Osim korištenja Gaussovog filtra, postoje još neke metode za filtriranje slike. Neke od njih koje OpenCV podržava uz Gaussov su: *medijan filter* te *bilateralni filter*.



Slika 10. Primjer glaćenja korištenjem Gaussovog filtra

Medijan filter se koristi na način da prolazi kroz svaki element, u ovom slučaju slike, te zamjeni svaki piksel sa srednjom vrijednošću susjednih piksela, koji su kvadratično od piksela koji se obrađuje. Kod malih i umjerenih zastupljenosti Gaussovog

šuma, medijan filter se pokazao puno boljim nego Gaussov filter kod uklanjanja šuma, uz očuvanje rubova slike.



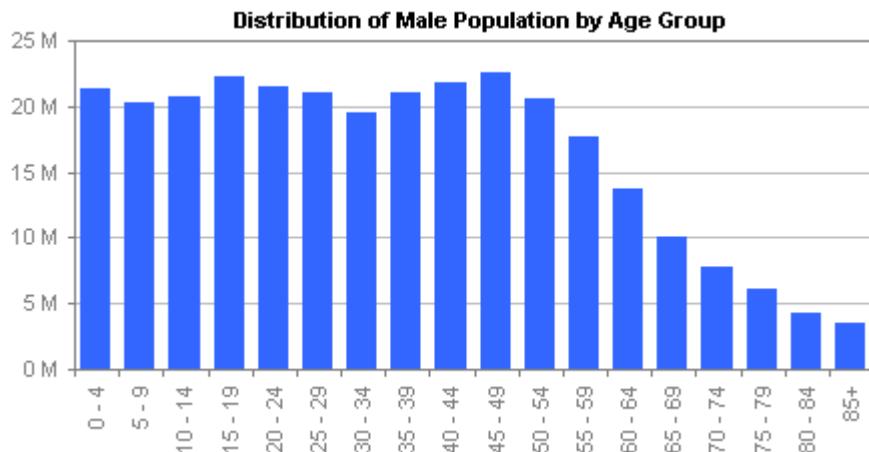
Slika 11. Primjer glađenja korištenjem medijan filtra

Bilateralni filter se koristi za očuvanje rubova prilikom uklanjanja šumova. Na analogan način kao i kod Gaussovog filtra, bilateralni filter također dodjeljuje susjednim pikselima težine. Te težine imaju dvije komponente, prva komponenta je jednaka kao i kod Gaussovog filtra. Druga komponenta uzima u obzir razliku u intenzitetu susjednih piksela i one koja se obrađuje.

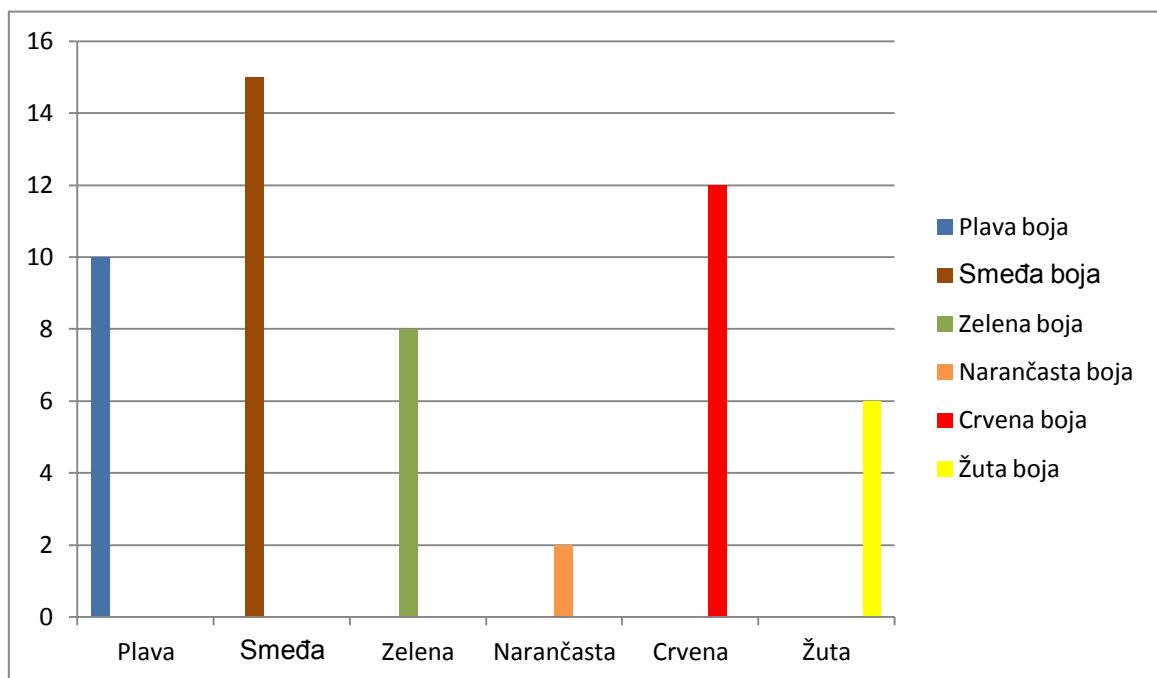
Uz gore navedene načine uklanjanja smetnji sa slike, postoje još neke druge metode kako ukloniti šum. Jedna takva metoda je izjednačavanje histograma.

Postoji više vrsta histograma. Općenito za histogram vrijedi da je on grafički prikaz pojavljivanja promatranih informacija izražen frekvencijom pojavljivanja. Na slici 12. se može vidjeti histogram dobi muške populacije, s donje strane su označene godine dok se s lijeve strane nalaze brojke koje predstavljaju broj ljudi. Ovakav prikaz omogućuje očitanje broja ljudi u određenim godinama, npr. na danoj slici možemo očitati da ima oko 20 milijuna ljudi koji su stariji od 30 a mlađi od 35 godina. U dalnjem tekstu će se govoriti samo o korištenom histogramu, histogramu boja. Histogram boja reprezentira raspodjelu boja na nekoj slici. Izveden je na način da prikazuje broj prebrojanih piksela koji pripadaju

određenim rasponima boja u dvodimenzionalnom ili trodimenzionalnom prostoru boja. Kao što je prikazano u histogramu boja.

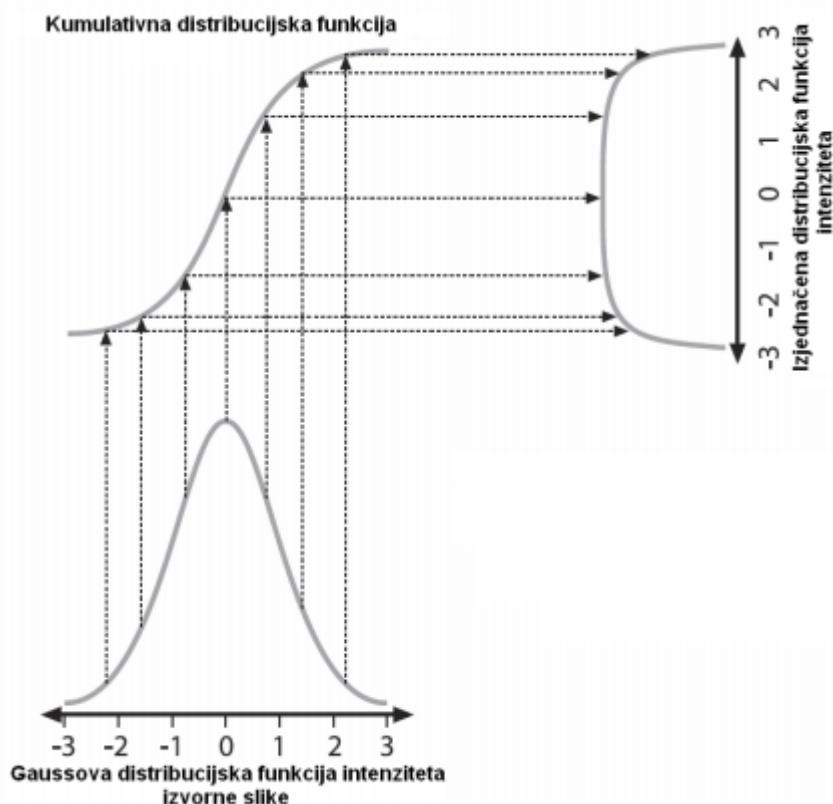


Slika 12. Histogram muške populacije [3]



Histogram boja

Izjednačavanje histogramom. Obzirom na vremenske uvjete u kojima se vrši detekcija, razlikujemo dobre i loše. Pod pojmom dobrih uvjeta smatramo uvjete u kojima je dobro izražen kontrast slike tj., ravnomjerna je distribucija intenziteta piksela na interval $[0, 255]$, pa je crvena boja znaka dovoljno dobro izražena na izvornoj slici. Tu spadaju primjerice slike snimane u sunčanim vremenskim uvjetima gdje imamo dobro osvjetljenje prometnih znakova. Pod pojmom loših uvjeta smatramo uvjete u kojima je loš kontrast slike tj., distribucija intenziteta ulazne slike nalazi se oko sredine intervala vrijednosti $[0, 255]$, pa je izvorna slika presvjetla ili suviše tamna. Tu spadaju primjerice slike snimane u sumrak ili u kišnim i maglovitim uvjetima gdje neke boje nisu dovoljno dobro izražene. Ukoliko se radi o lošim vremenskim uvjetima, da bi se ulazna slika poboljšala provodi se postupak izjednačavanja histograma. Cijela ideja iza navedenog postupka leži u tome da se loša poetna distribucija intenziteta izvorne slike raspoređuje u neku drugu širu i u idealnom slučaju uniformnu distribuciju intenziteta. Drugim riječima želimo zgušnute intenzitete izvorne slike ravnomjerno raspršiti na intervalu $[0, 255]$. To je moguće ukoliko je funkcija raspodijele kumulativna distribucijska funkcija. Radi pojašnjenja postupka uzmimo jednostavan primjer prikazan na slici 13. u kojemu se funkcija intenziteta izvorne slike ravna po Gaussovoj normalnoj razdiobi. Prikazanu kumulativnu distribucijsku funkciju u primjeru možemo iskoristiti ne samo na Gaussovoj, već i na bilo kojoj drugoj funkciji razdiobe intenziteta. Djelovanjem kumulativne distribucijske funkcije nad Gaussovom distribucijskom funkcijom intenziteta dobivamo izjednačenu distribucijsku funkciju intenziteta, tj. obavili smo izjednačavanje histograma. Izjednačavanje histograma unutar ostvarenog sustava provodi se tako da postupak primjenjujemo na svaki RGB kanal posebno, pri čemu je svaki kanal predstavljen crno bijelom slikom.

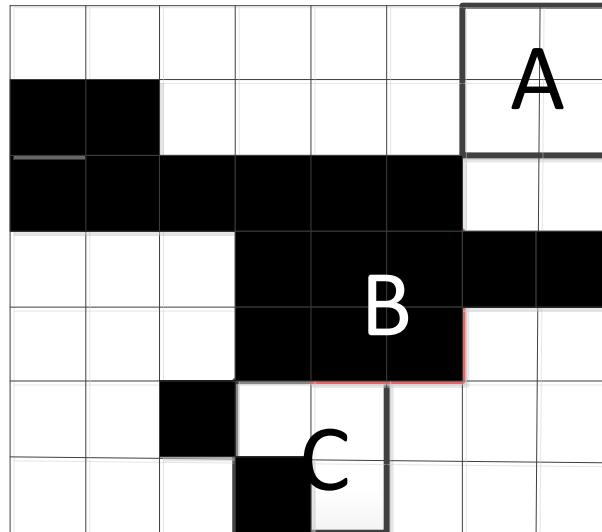


Slika 13. Primjer izjednačavanja histograma

Međutim ova metoda se neće koristiti u izradi ovog diplomskog rada. Već je samo spomenuta kao jedan od načina kako se može eliminirati šum. Međutim, ovakva metoda reduciranja šuma, i nije najbolje optimalna kada se objekt nalazi u vodi, to jest, kada su slike nastale pod vodom. Dosta je smanjeno razlučivanje boja, te najčešće i prepreke poprimaju sličnu boju kao i objekt koji se želi detektirati. Osim te metode, postoje još dvije koje će se koristiti u implementaciji algoritma za detekciju. Te dvije metode su: *erodacija* i *dilatacija*, spadaju u morfološku obradu slike.

Morfološka obrada slike je skup nelinearnih vezanih operacija za oblik ili morfologiju značajki na slici. Morfološke operacije ovise o relativnoj vrijednosti piksela, a ne na njihovim brojčanim vrijednostima, i stoga su posebno pogodna za obradu binarnih slika. Morfološka tehnika uspoređuje sliku s malim oblikom koji se zove strukturirajući element. Strukturirajući element je smješten na svim mogućim pozicijama u slici i to se uspoređuje s istim susjednim pikselima, na slici 14. predstavlja okvir A. Tokom

uspoređivanja strukturirajući element se "uklapa" sa susjednim pikselima slike, okvir B, ili "pogodi" to jest siječe susjedne piksele slike, okvir C.



Slika 14. Primjer morfološke obrade

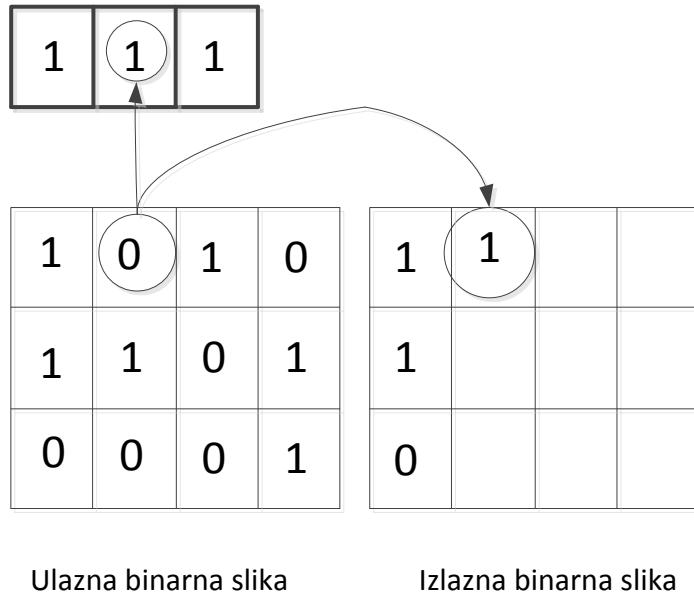
Morfološke operacije nad binarne slike stvara novu binarnu sliku u kojoj piksel ima vrijednost različitu od nule samo ako je test uspješan u tom mjestu u ulaznoj slici, to jest, sa slike 14. okviri B i C. Strukturirajući element je mala binarna slika, to jest, mala matrica piksela, sa vrijednostima nula ili jedan. Strukturirajući element igra istu ulogu kao i konvolucija kod filtara.

Erozija radi na način da postavi vrijednost izlaznog piksela na *minimalnu* vrijednost od svih susjednih vrijednosti piksela ulazne slike. Kod binarnih slika, ako je barem jedan od susjednih piksela postavljen na vrijednost 0, tada je i izlazna vrijednost piksela postavljena na 0.

Dilatacija radi tako što postavi vrijednost izlaznog piksela na maksimalnu vrijednost piksela od svih susjednih vrijednosti piksela ulazne slike. Kad je riječ o binarnoj slici, ako je barem jedan od susjednih piksela postavljen na vrijednost 1, tada je i izlazna vrijednost piksela postavljena na 1.

Na slijedećoj slici je prikazan način rada erozije i dilatacije.

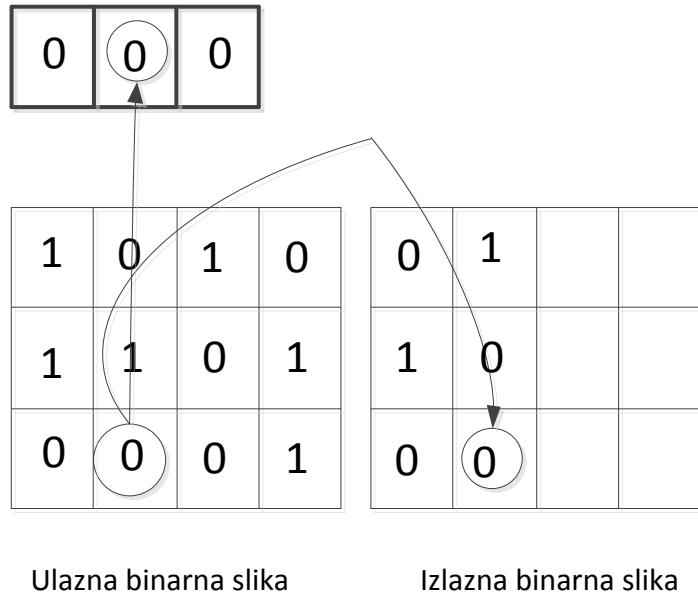
Strukturirajući element



Slika 15. Prikaz primjera korištenja dilatacije

Sa slike 11. jasno se vidi da je strukturirajući element, to jest okvir, poklopio prvo polje dimenzije 1x3. Kao promatrani piksel se gleda zaokružena na prethodnoj slici, koristeći pravilo za dilataciju, jasno se vidi da su 2 susjedna piksela sa vrijednostima 1, što znači da će izlazna vrijednosti piksela bit postavljena na 1, umjesto vrijednosti 0 kako je bilo zadano na ulaznoj binarnoj slici.

Strukturirajući element



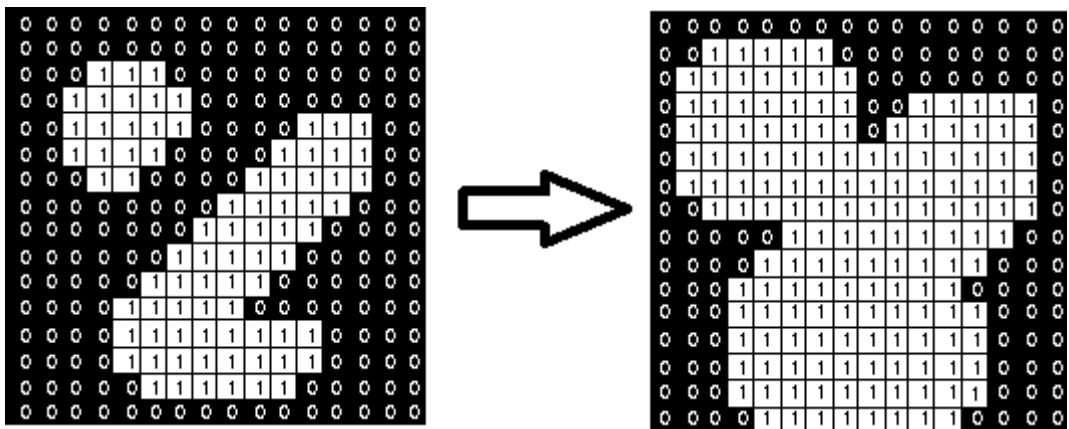
Ulagna binarna slika

Izlazna binarna slika

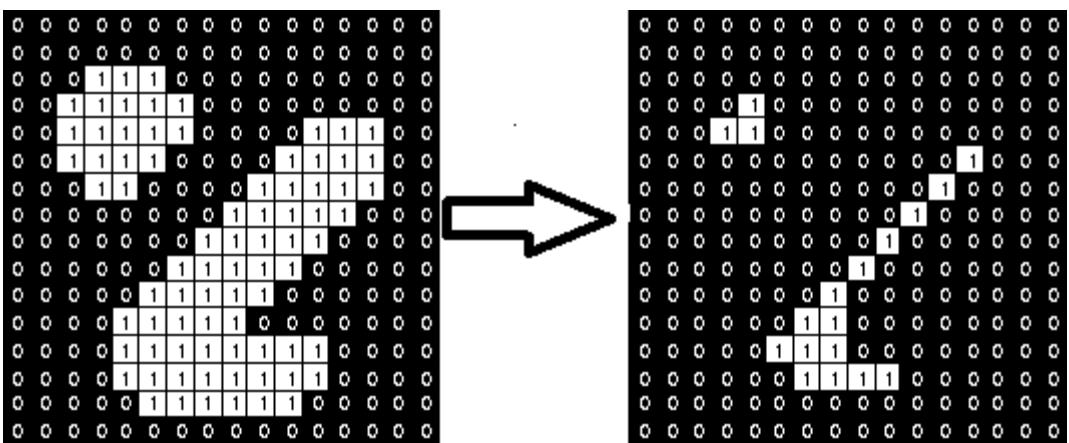
Slika 16. Prikaz primjene postupka erozije

Na slici 12. se može uočiti da je strukturirajući element, to jest, okvir, poklopio zadnje polje dimenzije 1x3, te poštujući pravilo za eroziju, vrijednost izlaznog piksela je 0. Za zorniji prikaz pravila erozije, može se pogledati 2.red, polje (1,1,0). Nakon primjene pravila za eroziju, izlazna vrijednost piksela se postavi na 0, umjesto vrijednosti 1.

U suštini, dilatacija širi regiju, popunjava male šupljine u binarnoj slici, to jest, vrijednost 0, što predstavlja crnu boju, zamijeni sa vrijednosti 1, što predstavlja bijelu boju. Dok erozija smanjuje željenu regiju, tako što uklanja manje šupljine. Odnosno, piksele s vrijednostima 1 (bijela boja), zamjenjuje s vrijednostima piksela 0 (crna boja). Što se jasno vidi na slijedećim slikama (slika 17. i slika 18.).



Slika 17. Prikaz dilatacije [4]



Slika 18. Prikaz erozije [5]



Slika 19. Prikaz utjecaj erozije na binarnu sliku



Slika 20. Prikaz utjecaja dilatacije na binarnu sliku

2.1.1.3 Adaptivni prag

Izloženost svjetlosti i sjeni, može uzrokovati zamagljivanje bitnih informacija u slici, nastala zbog loših uvjeta rasvjete. Globalna tehnika praga se koristiti za segmentiranje piksela u bijele ili crne piksele pomoću unaprijed definiranih vrijednosti, adaptivni prag se koristi za odvajanje željenih objekata u prednjem planu iz pozadine. Ova metoda je prikladna za slike s jakim gradijentom osvjetljenja, budući da objekti u nisko/visoko osvijetljenim mjestima neće nestati kao što je slučaj kod normalnog određivanje praga. Adaptivni prag rješava te probleme pomoću, koja je u suprotnosti s globalnom metodom praga, tako da se postavi individualna vrijednost praga za svaki piksel u slici. Izračun uključuje vrijednosti piksela oko piksela koji se obrađuje, na način da se korištenju konvolucijski operatera bilo srednja vrijednost ili medijan.

Algoritam 2. Adaptivni prag

Četiri koraka opisuju načelo rada adaptivnog praga.

1. Koristiti konvoluciju nad slikom, koristeći prikladne statističke operatore, kao što su srednja vrijednost ili medijan.
2. Razdvojiti originalnu sliku od one „pročešljane“ konvolucijom.
3. Postaviti prag na konstantnu vrijednost te proći kroz razlike u slici.
4. Napraviti inverz slike nakon prethodne operacije.



Slika 21. Usporedba normalne tehnike (lijeva slika) te adaptivne tehnike (desna slika)

Jasno se vidi iz slike 21. da je adaptivni prag bolja tehnika, nego što je to slučaj sa normalnim pragom. Jasnije se ocrtavaju rešetke kaveza na desnoj slici, to jest, na kojoj je primijenjena adaptivna tehnika praga.

2.1.1.4 Pronalaženje vanjskih kontura

Kontura je skup točaka koje predstavljaju krivulje u slikama. Odnosno opisuju rubove objekata u binarnim slikama. Jednostavna ilustracija spajanja kontura. Neprekinuti lanci rubnih piksela stvaraju se rekurzivno tako da se funkcija koja sprema piksele lanca u memoriju poziva za susjedne piksele pronađenog piksela ruba. Ukoliko je jedan od susjeda promatranog piksela također dio ruba tada se funkcija dalje poziva za njegove susjede i tako sve dok za trenutni piksel postoje susjedni pikseli ruba, odnosno svaki piksel sadrži informaciju o povezanosti sa susjednim pikselom. Ukoliko je broj piksela pronađenog lanca manji od zadanog praga tada se taj lanac odbacuje. Metoda kojom se nalaze konture, korištena u bibliotekama OpenCv-a je Suzuki 85 [6]. Konture se korisni alata za pronalaženje i detekciju objekata, uz poznati vanjski izgled ciljanog objekta.

Algoritam 3. Suzuki 85.

Neka ulazna slika bude zapisa $F=\{f_{i,j}\}$. Postavljanje NBD na inicijalnu vrijednost 1. NBD (sekvencijalni broj trenutne granice/ruba). Proći po cijeloj slici koristeći Tv raster, te za svaki piksel za koji vrijedi da je $f_{i,j} \neq 0$, primijeniti slijedeće korake:

- 1.) a) Ako je $f_{i,j}=1$ i $f_{i,j-1}=0$, tada se uzima da je piksel (i,j) granica/rub koja prati od početne točke vanjsku granicu/ruba, NBD se povećava, te $(i_2,j_2) \leftarrow (i,j-1)$

- b) Inače ako, $f_{i,j} \geq 1$ i $f_{i,j+1}=0$, tada se uzima da je piksel (i,j) granica/rub koja prati od početne točke unutarnju granicu/ruba, povećava se NBD, te
 $(i_2,j_2) \leftarrow (i,j+1)$ i LNBD $\leftarrow f_{i,j}$ u slučaju da je $f_{i,j} > 1$.
- c) inače pređi na korak 4.).

- 2.) Ovisno o tipu nove granice/rubovi ili granice sa sekvencijskim brojem LNBD, određuje se roditelj prema tablici 2.

- 3.) Počevši od startne točke (i,j) , prate se detektirane granice/rubovi, koristeći se slijedećim koracima:

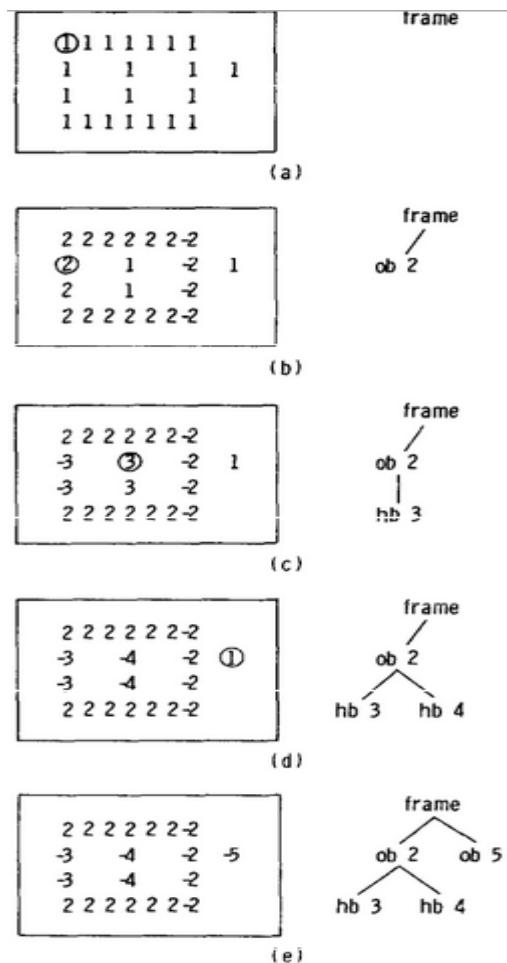
- a) Kreni od točke (i_2,j_2) , proći kroz susjedne piksele od (i,j) u smjeru kazaljke na satu te naći piksele sa vrijednostima različitim od 0. Neka je (i_1,j_1) prvi pronađeni piksel različit od 0. Ako nema piksela različitih od 0, postavi NBD na $f_{i,j}$ i pređi na korak 4.).
- b) $(i_2,j_2) \leftarrow (i_1,j_1)$ te $(i_3,j_3) \leftarrow (i,j)$.
- c) Kreni od točke (i_2,j_2) , i u smjeru suprotnom od smjera kazaljke na satu, prođi kroz susjedne piksele od točke (i_3,j_3) te nađi piksele s vrijednosti različitom od 0 te je postavi kao slijedeću točku (i_4,j_4) .
- d) Promjeni vrijednost f_{i_3,j_3} od piksela na poziciji (i_3,j_3) na slijedeće načine:

- | | |
|--|---|
| | <ol style="list-style-type: none"> 1.) Ako je piksel (i_3,j_3+1) 0-piksel, tada $f_{i_3,j_3} \leftarrow -\text{NBD}$. 2.) Ako je piksel (i_3,j_3+1) različit od 0, te $f_{i_3,j_3} = 1$, tada $f_{i_3,j_3} \leftarrow \text{NBD}$. |
|--|---|

3.) Inače ne radi promjene kod f_{i_3,j_3} .

- e) Ako je $(i_4, j_4) = (i, j)$ i $(i_3, j_3) = (i_1, j_1)$, vratili se u početnu točku, idi na korak 4.), inače $(i_2, j_2) \leftarrow (i_3, j_3)$, $(i_3, j_3) \leftarrow (i_4, j_4)$, te se vrati na korak 3 a).

- 4.) Ako je $f_{i,j} \neq 1$, tada je $\text{LNBD} \leftarrow |f_{i,j}|$, te nastavi rastersko skeniranje od piksela $(i, j+1)$. Algoritam završava kada dođe do donjeg desnog kuta slike.



Slika 22. Ilustracija rada Suzuki 85 algoritma.[7]

S lijeve strane su prikazani vrijednosti piksela od zadane slike, s desne strane su izvučene strukture duž granice/ruba, (ob- outline border, vanjski rubovi, hb- hole border, unutarnji rubovi). Zaokruženi pikseli predstavljaju početne točke za praćene granice/ruba.



Slika 23. Primjer detekcije vanjski kontura

2.1.1.5 Površina kontura

Kao još jedan oblik filtracije smetnji u slici kod detekcije pravilne strukture kaveza za uzgoj ribe, koristio se uvjet površine nađenih kontura. Površina konture računa koristeći Greenov teorem o zatvorenoj krivulji. Kao dodatan uvjet uklanjanja smetnji od pravilnih struktura, koristio se uvjet konveksnosti. To jest, da je svaki kut u četverokutu manji od 180° . Više o Greenovom teoremu na [8].

Algoritam 4. Površina zatvorenih kontura

- 1.) Nađi zatvorenu konturu,
 - a) Ako je površina zatvorene konture unutar zadanih granica, te vrijedi uvjet konveksnosti konture, tada dana kontura predstavlja objekt koji želimo detektirati.
 - b) Inače, zatvorena kontura ne predstavlja objekt koji želimo detektirati, onda prijeđi dalje na slijedeću zatvorenu konturu, to jest, ponovo na korak 1. Algoritam traje dok se ne prođu sve zatvorene konture sa ulazne slike.

3. Programska implementacija

3.1. Implementacija za obradu slike

Algoritam detekcije pravilne strukture kaveza za uzgoj riba razvijen je u Microsoft Visual Studio-u programskom paketu, uz korištenje OpenCV biblioteke, koje je u prethodnom poglavlju opisano kako ga „instalirati“, pri tome se koristio C kao programski jezik.

Implementacija algoritma za detekciju pravilne strukture kaveza za uzgoj ribe se odvija prema shemi sa slike 8. Odnosno algoritam se može podijeliti na 3 dijela, prvi dio predstavlja ulazna slika, drugi dio obradu slike, te u konačnici detekcija. Obrada slike se sastoji od koraka binarizacije slike, glađenja, adaptivnog praga te pronalaženja vanjskih kotura/rubnica na slici.

Prvi korak je učitavanje slike. A to se radi na slijedeći način,

```
IplImage* img = cvLoadImage("C:/Users/roko romic/Desktop/fish.png");
```

Funkcija *cvLoadImage()* je rutina visoke razine, koja određuje format podatka koji se učitava na temelju imena podatka. Ona automatski alocira memoriju računala u koju sprema strukturu podataka slike. Te vraća pokazivač na alociranu memoriju preko imena *img*.

Nakon što se učitala slika, slijedi korak pretvaranja slike u sliku sa sivim tonovima (eng. grayscale).

```
IplImage* imgGrayScale = cvCreateImage(cvGetSize(img), 8, 1);  
cvCvtColor(img, imgGrayScale, CV_BGR2GRAY);
```

metodom *cvCreateImage()* inicijalizira se te postavi zaglavlj. Pri čemu je prvi argument veličina slike, drugi argument je dubina te zadnji broj kanala slike. Pretvaranje slike u sliku sa sivim tonovima radi se preko metode *cvCvtColor()*, tako da se postavi treći argument na Cv_BGR2GRAY, što predstavlja kodno ime za određenu konverziju. Dok je prvi argument pokazivač na izvornu sliku, a drugi na izlaznu, nakon konverzije.

Nakon konverzije slijedi glađenje. Postiže se tako da se pozove slijedeća metoda *cvSmooth()*.

```
cvSmooth(imgGrayScale, imgGrayScale, CV_MEDIAN,3);
```

prvi argument je izvorna slika, drugi je izlazna nakon glađenja, to jest pokazivač na memoriju, te treći argument predstavlja tip filtra koji se koristi. Postoje tri načina filtriranja dostupna kod metode glađenje. Medijan filter, Gaussov filter te bilateralni filter. Postavljaju se tako da se postavi odgovarajuće kodno ime. Tako za medijan filter CV_MEDIAN, za Gaussov filter CV_GAUSSIAN, te za bilateralni filter CV_BILATERAL.

```
cvSmooth(mf, mf, CV_GAUSSIAN,5,5);
```

Daljni korak je adaptivni prag, koji postavlja adaptivni prag na polje podataka. Koristi se slijedeća metoda *cvAdaptiveThreshold()*.

```
cvAdaptiveThreshold(imgGrayScale, Iat, 255, CV_ADAPTIVE_THRESH_MEAN_C,  
CV_THRESH_BINARY, 71, 15);
```

Sadrži više ulaznih parametara. Prvi je ulazna slika, drugi izlazna slika nakon obrade, treći parametar je maksimalna vrijednost [1,255] koju može poprimiti piksel ako zadovolji uvjete, četvrti parametar je adaptivna metoda koja se koristi. Postoje dvije, jedna je *ADAPTIVE_THRESH_MEAN_C*, a druga je *ADAPTIVE_THRESH_GAUSSIAN_C*. Kod *ADAPTIVE_THRESH_MEAN_C* metode vrijednost praga $T(x,y)$ je srednja vrijednost od veličine bloka x veličina bloka susjednih točaka oko (x,y) te se oduzme sa parametrom C. Dok kod *ADAPTIVE_THRESH_GAUSSIAN_C* vrijednost praga $T(x,y)$ je otežana suma veličina bloka x veličina bloka susjednih točaka oko (x,y) umanjena za parametar C. Peti parametar je tip praga, *THRESH_BINARY* ili *THRESH_BINARY_INV*. Za prvi vrijedi $dst(x,y)=max$ vrijednost (vrijednost trećeg ulaznog parametra), ako je $src(x,y)>T(x,y)$, inače je $dst(x,y)=0$. Za drugi vrijedi $dst(x,y)=0$, ako je $src(x,y)>T(x,y)$, inače je $dst(x,y)=max$ vrijednost.

Daljnji koraci su erozija i dilatacija. Jednostavno se ostvare pozivanjem slijedećih metoda.

```
cvErode(Iat,Iat,NULL,2);  
cvDilate(Iat,Iat,NULL,2);
```

Prvi argument predstavlja ulaznu sliku, drugi izlaznu nakon obrade, treći argument predstavlja veličinu polja oko piksela da uzima, ako je postavljena vrijednost NULL, tada

je veličina polja 3x3, te posljednji parametar broj iteracija u kojima se provodi erozija/dilatacija.

Slijedeći korak je traženje vanjskih kontura. To se postiže pozivanjem slijedeće metode *cvFindContours()*.

```
CvSeq* contours; //postavlja pokazivač na koloturu u memoriju
CvSeq* result; //sadrži sekvence točaka konture
CvMemStorage *storage = cvCreateMemStorage(0); //prostor u koji će se
spremati konture

//pronalaženje svih kontura u slici
cvFindContours(Iat, storage, &contours, sizeof(CvContour), CV_RETR_LIST,
CV_CHAIN_APPROX_SIMPLE, cvPoint(0,0));
```

Svaka kontura je spremljena kao polje sa točkama. Prvi argument je slika koja se prethodno provukla kroz sve metode, drugi argument predstavlja memorijski prostor u koji se pohranjuju konture, treći je veličina zaglavlja, četvrti argument predstavlja način pronalaženja kontura. Postoji nekoliko vrsta, *CV_RETR_EXTERNAL*, *CV_RETR_LIST*, *CV_RETR_CCOMP* te *CV_RETR_TREE*. Slijedeći argument su aproksimacijske metode. Postoje *CV_CHAIN_APPROX_NONE*, *CV_CHAIN_APPROX_SIMPLE* te *CV_CHAIN_APPROX_TC89_L1*. Zadnji argument je odmak od trenutnog piksela. Zadnji dio implementacije je da se prođe po svim konturama, te ako je površina konture unutar zadanih granica, te vrijedi da je kontura konveksna, što se provjerava sa slijedećom metodom. *cvCheckContourConvexity()*, te iscrta se nađena kontura.

```
while(contours)
{
    area = cvContourArea( contours, CV_WHOLE_SEQ ); //računa se površina
    zatvorenih kontura

    //sprema se točke kontre u result
    result = cvApproxPoly(contours, sizeof(CvContour), storage, CV_POLY_APPROX_DP,
    cvContourPerimeter(contours)*0.02, 0);

    if( fabs( area ) > pix && fabs( area ) < 10000 && cvCheckContourConvexity(result))
    {
        if(result->total==3) //ako sadrži tri točke (trokut)
        {
            //iterira se kroz sve točke
            CvPoint *pt[3];
            for(int i=0;i<3;i++){
                pt[i] = (CvPoint*)cvGetSeqElem(result, i);
            }
        }
    }
}
```

```

//crtanje linija
cvLine(img, *pt[0], *pt[1], cvScalar(255,0,0),4);
cvLine(img, *pt[1], *pt[2], cvScalar(255,0,0),4);
cvLine(img, *pt[2], *pt[0], cvScalar(255,0,0),4);

}

//ako sadrži četiri točke kontura
else if(result->total==4 )
{
    //iteracija kroz sve točke
    CvPoint *pt[4];
    for(int i=0;i<4;i++){
        pt[i] = (CvPoint*)cvGetSeqElem(result, i);
    }

    //crtanje linija u obliku četverokuta te obojiti liniju u zeleno (scScalar)
    cvLine(img, *pt[0], *pt[1], cvScalar(0,255,0),4);
    cvLine(img, *pt[1], *pt[2], cvScalar(0,255,0),4);
    cvLine(img, *pt[2], *pt[3], cvScalar(0,255,0),4);
    cvLine(img, *pt[3], *pt[0], cvScalar(0,255,0),4);
}
}
//dohvaćanje sljedeće konture
contours = contours->h_next;
}

```

Na kraju je potrebno počistiti alociranu memoriju, a to se postiže na sljedeći način.

```

cvDestroyAllWindows();
cvReleaseMemStorage(&storage);
cvReleaseImage(&img);

```

3.2. Implementacija za obradu video slike

Sve navedeno kod implementacije za obrade slike vrijedi i za obradu video slike. Odnosno video se podijeli na sličice (eng. frame), te se obrađuje svaka sličica na način opisan u prethodnom poglavlju. Potrebno je malo doraditi kod, tako da radi za ulazne podatke kao što je video.

```

CvCapture *capture = cvCaptureFromAVI(argv[1]);

int frame_width=(int) cvGetCaptureProperty(capture,
CV_CAP_PROP_FRAME_WIDTH);      //dohvaća se širina okvira

int frame_height=(int) cvGetCaptureProperty(capture,
CV_CAP_PROP_FRAME_HEIGHT);     //dohvaća se visina okvira

```

```

        int fps = ( int )cvGetCaptureProperty( capture, CV_CAP_PROP_FPS );
//dohvaća se broj okvira u sekundi (eng. fps)
    //stvaranje strukture za spremanje videa
    CvVideoWriter *writer = 0;

    writer=cvCreateVideoWriter(argv[2],CV_FOURCC('M', 'P', '4',
'2'),fps, cvSize(frame_width,frame_height),1);

```

Metoda *cvCaptureFromAVI()* služi za dohvaćanje video sadržaja te ga pohraniti u memoriju. Metodom *cvQueryFrame()* se dohvaća svaki okvir zadanog videa.

```
IplImage* frame = cvQueryFrame( capture );
```

Metoda *cvCreateVideoWriter()* kao parametre ima, ime videa koji želimo spremiti, kodeci u kojem želimo spremiť video, broj okvira u sekundi, te veličina okvira.

Kao razlog ubrzanja obrade videa, a samim time i veličinu izlaznog video materijala, implementiran je dio za preskakanje pojedinih sličica. Realiziran je na način da se uvede brojač sličica, te se provede operacija *modulo (%)* nad brojačem. Ukoliko ostatak te operacije zadovoljava postavljene uvjete provodi se obrada trenutne sličice. Dio algoritma jer prikazan ispod.

```

// frame counter
    int brojac=0;
//frame skip counter
    int skipfr;

while(true) {
    //dohvaćaj sekvencijalno svaku sličicu
    IplImage* frame = cvQueryFrame( capture );
        brojac=brojac+1;
        printf("%d\n",brojac);
        if (brojac%skipfr==0){ // uzima u obzir svaku skipfr
sličicu
            ...
}

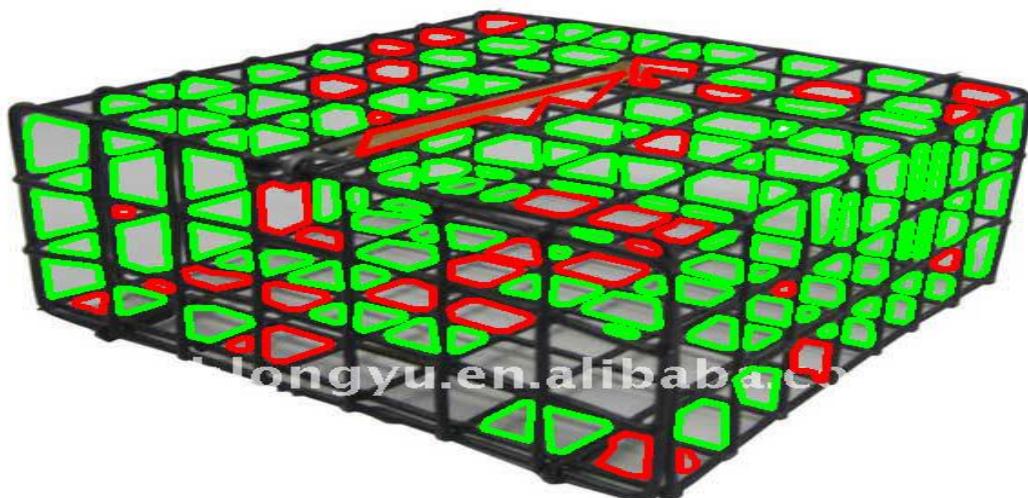
```

4. Eksperimentalni rezultati

U ovom poglavlju su prikazani rezultati detekcije pravilne strukture kaveza za uzgoj riba. Kako se video ne može prikazati u ovom radu, rezultati detekcije će biti prikazani u obliku slika. Ulazne i izlazne slike su formata *PNG*. Rezultate detekcije su podijeljeni u tri skupine. Uspješna detekcija, djelomična detekcija te neuspješna detekcija. Pod uspješnom detekcijom smatra se da su pronađene sve pravilne strukture na slici. Za djelomičnu detekciju ako je detektiran dio pravilnih struktura, dok za neuspješnu detekciju se smatra ako nije detektirana pravilna struktura na slici.

4.1. Primjeri uspješne detekcije

Da bi detekcija bila uspješna, slika mora biti dovoljno dobre kvalitete. Osim toga, značaju ulogu igra i osvjetljenje, udaljenosti s koje je sniman objekt detekcije te utjecaj fizičkih prepreka. Korištenjem prethodnog opisanog algoritma, neke se situacije mogu pokazati dobrim za detekciju, ali u većini slučaja to neće biti tako. Kao primjeri koristit će se slike dostupne na internetu.



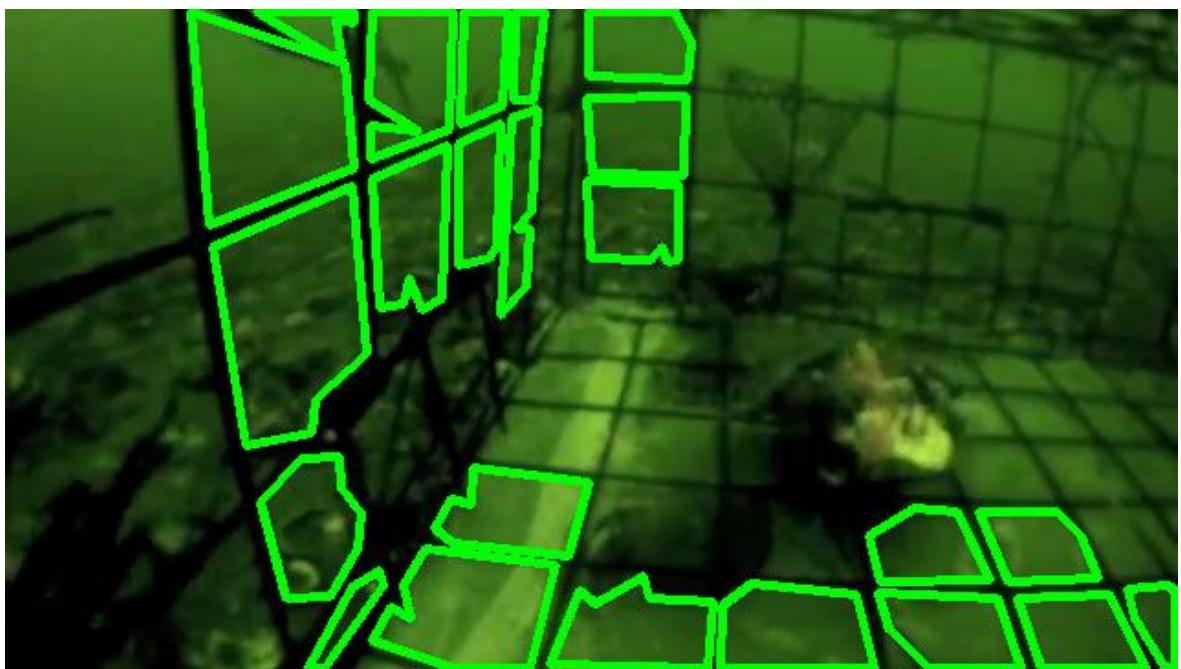
Slika 24. Primjer detekcije pravilne strukture kaveza



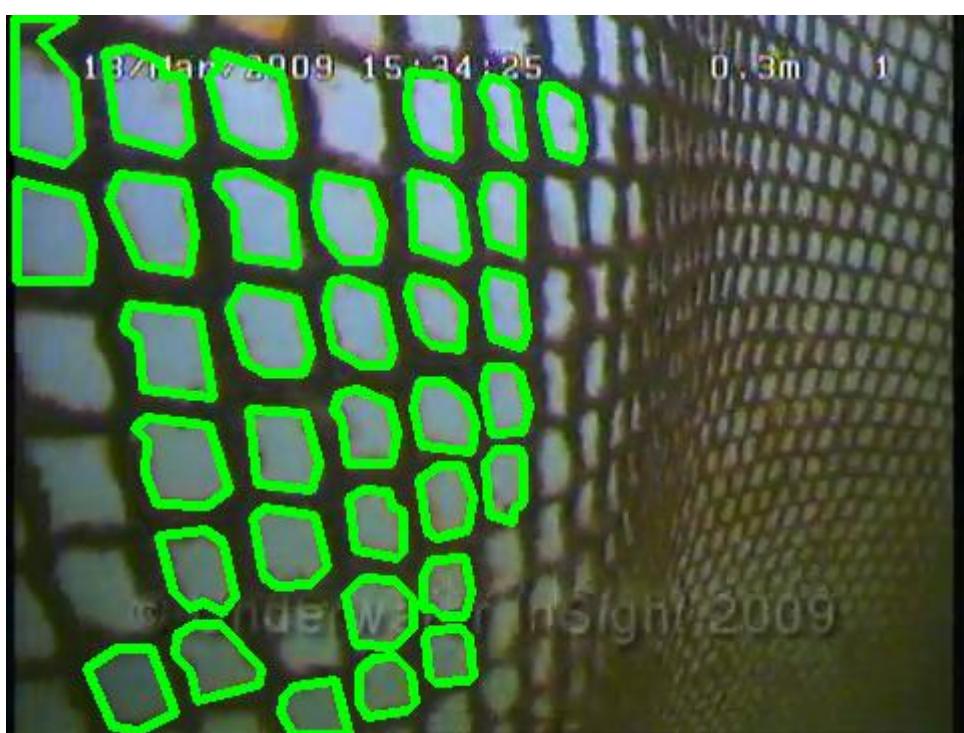
Slika 25. Primjeri detekcije pravilne strukture kaveza



Slika 26. Izvučena sličica iz obrađenog video zapisa



Slika 27. Izvučena sličica iz obrađenog video zapisa



Slika 28. Izvučena sličica iz obrađenog video zapisa



Slika 29. Izvučene sličice iz obrađenog video zapisa, uz korištenje uvjeta konveksnosti

Iz prethodnik slika se jasno vidi kako utjeću već ranije navedeni problemi koji se javljaju kod detekcije. Prvenstveno se odnosi na osvjetljenost te kvaliteti slike, odnosno sličice izvučene iz video zapisa (slike 26. i 27.). Na slici 25. s lijeve strane, se osim detektiranih rešetki vidi i detektirani dijelovi koji su nastali spajanjem rešetki i ribe. To je isto jedan od problema koji se javlja zbog prisutnosti drugog objekta u kadru objekta kojeg se želi detektirati. Isti slučaj se vidi i na slikama 26. i 27. Ta se problem rješava korištenjem uvjeta konveksnosti zatvorne konture, rezultat je prikazan na slici 29. Iz slike 28. se može zaključiti kako detekcija ovisi i o udaljenosti objekta za detekciju.

4.2. Primjeri djelomično uspješne detekcije

U ovaj segment spadaju one detekcije koje detektiraju objekt, ali taj objekt nije u potpunosti rešetka, već sadrži i dio nekog drugog objekta ili je detektiran neki objekt koji nije željeni. Kao što je to prikazano na idućim slikama.



Slika 30. Primjeri djelomično uspješne detekcije

Osim rešetki, detektirani su i dijelovi ronioca, prikazano na slici 30. s desne strane. Ili dijelovi rešetke spojeni s dijelovima ribe, prikazano na slici 30. s lijeve strane. To se ukloni na način da se koristi uvjet konveksnosti.

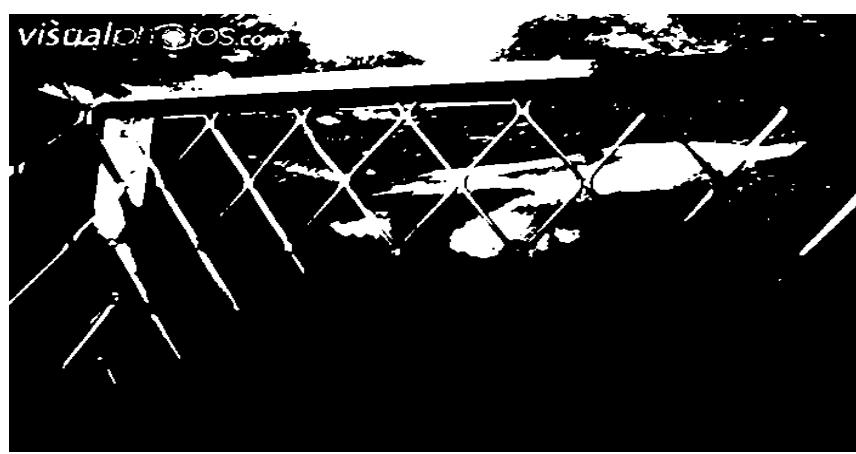
Usporedbom slike 30. te slike 25. (slike koje se nalaze s lijeve strane), jasno se može uočiti utjecaj veličine zatvorene konture koju algoritam pronađe, na određivanje kandidata detekcije. Na slici 25. kao donja granica za površinu kontura uzimala vrijednost 20, dok za sliku 30. ta ista granica je postavljena na 100.

4.3. Lažne detekcije

Lažne detekcije najčešće nastaju kada je omjer svjetla sjene, odnosno kontrasta u slici nepovoljan. Tada algoritam pronađe zatvorene konture, ali te konture ne odgovaraju konturama rešetki, već ostalim dijelovima slike. To se jasno vidi na slijedećoj slici.



Slika 31. Primjer loše detekcije



Slika 32. Binariziran izgled slike

Iz slike 31. vidi se kako kontrast slike utječe na detekciju.

5. Stereo vizija

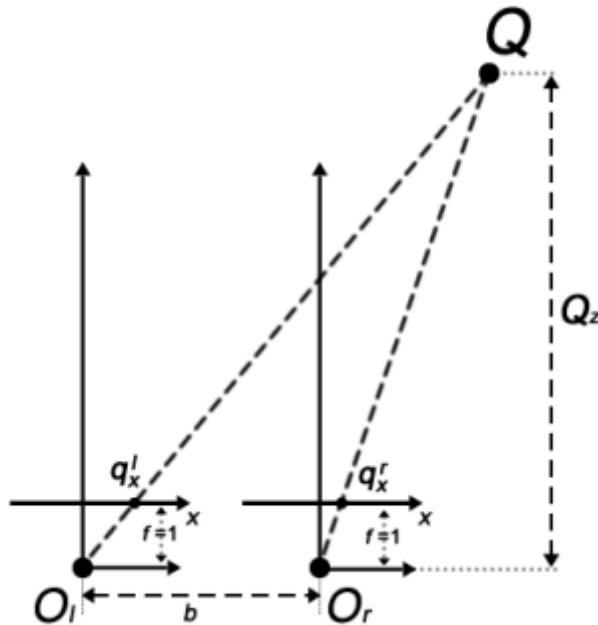
Dio ovog diplomskog rada je i istražiti mogućnost određivanje udaljenosti kamere od detektiranog objekta. Međutim kako je riječ o snimkama jedne kamere, jednostavno je nemoguće doći do dubine slike. Metoda kojom se dolazi do izračuna dubine slike, koristeći pri tome najmanje dvije kamere zove se stereo vizija. Stereo vid bazira se na pronalaženju podudarnosti između točaka slike lijeve i desne kamere. Uz poznati razmak između kamera (engl. baseline), moguće je izračunati koordinate točaka u 3D prostoru. Stereo vizija najčešće se provodi u četiri koraka:

1. Matematičko uklanjanje radikalne i tangencijalne distorzije leće (engl. undistortion),
2. Rektifikacija, odnosno projekcija slika dviju kamera u zajedničku slikovnu ravninu (engl. rectification). Nakon ovog koraka slike su poravnate po recima.
3. Pronalaženje korespondentnih značajki u lijevoj i desnoj slici (engl. correspondence). Nakon ovog koraka poznati su nam dispariteti značajki, odnosno razlike u iznosima x-koordinata značajki u lijevoj i desnoj slici: $x^l - x^r$.
4. Ukoliko su nam poznata geometrijska svojstva kamera, moguće je iz mape dispariteta izračunati udaljenosti točaka od kamere u 3D prostoru. Ovaj korak zove se reprojekcija (engl. reprojection), a njegov izlaz dubinska mapa (engl. depth map).

5.1. Triangulacija

Prepostavimo da imamo idealni stereo sustav, bez distorzije, horizontalno poravnat, kakav prikazuje slika 32. Slikovne ravnine obje kamere su međusobno koplanarne te imaju paralelne optičke osi, čiji razmak nam je poznat (sijeku se u beskonačnosti). Optička os (engl. optical axis, principal ray) je zraka koja izlazi iz centra projekcije O kroz osnovnu točku (engl. principal point) c. Osnovna točka je točka u kojoj optička os siječe slikovnu ravninu i ne mora nužno biti jednaka centru slike. Obje kamere

imaju jednake žarišne duljine (engl. focal length) $f_l=f_r=f=1$, te su im osnovne točke na istim koordinatama u svojim slikama.

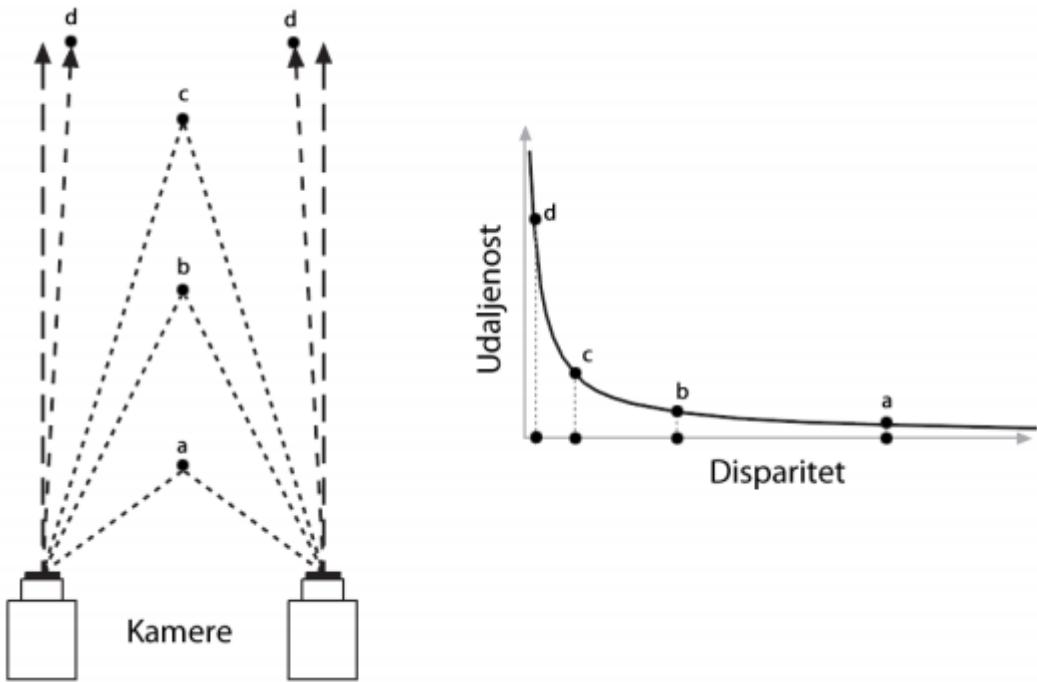


Slika 33. Idealni stereo sustav

Krećući od prepostavke da se točka Q iz stvarnog svijeta može pronaći u lijevoj i desnoj slici na pozicijama q_l i q_r , čije horizontalne koordinate su q_x^r i q_x^l . Slijedeći pravilo sličnosti trokuta, lako se pokaže da je dubina Q_z obrnuto proporcionalna disparitetu $d = q_x^r - q_x^l$, što je prikazano sljedećim jednadžbama:

$$\begin{aligned} \frac{Q_x^l}{Q_z} &= q_{lx}, \quad \frac{Q_x^r}{Q_z} = q_{rx} \\ \frac{Q_x^l - b}{Q_z} &= q_{rx}, \\ q_x^l - q_x^r &= \frac{Q_x^l - Q_x^r + b}{Q_z} \Rightarrow Q_z = \frac{b}{d} \end{aligned} \quad (2)$$

Zahvaljujući toj činjenici, uz prepostavku da je udaljenost između kamera fiksna, kada je disparitet blizu nule, male promjene u disparitetu znače velike promjene u dubini, dok u slučaju kada je disparitet velik, male promjene u disparitetu ne utječu znatno na promjenu dubine. Direktna posljedica toga je da stereo sustavi postižu dobru dubinsku rezoluciju samo za objekte koji se nalaze relativno blizu kameri, što prikazuje slika 33.

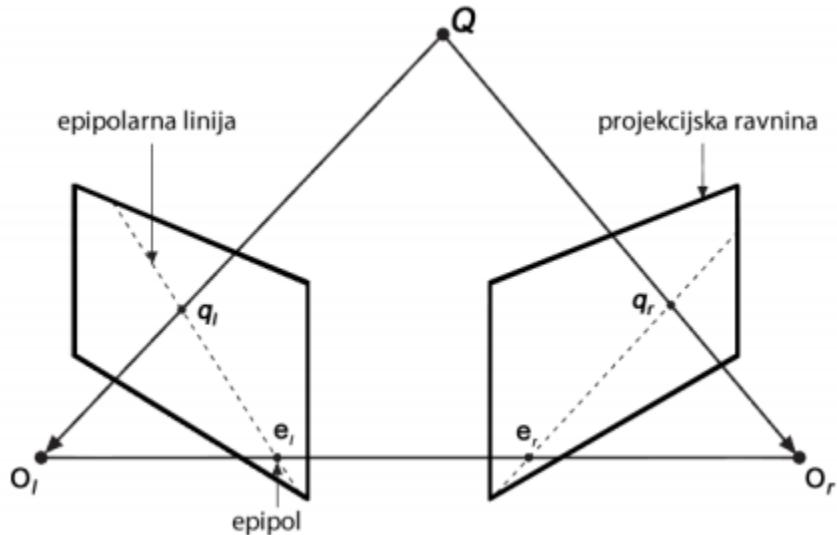


Slika 34. Obrnuto proporcionalnost dispariteta i dubine

Također, vrlo je bitno i da kamere budu vremenski sinkronizirane, kako bismo izbjegli probleme s pomicnim objektima u sceni (kao i pomicnim kamerama). U suprotnom smo limitirani na korištenje stacionarnih kamera za stacionarne scene. S obzirom da se mnoge operacije pojednostavljaju u idealnom rasporedu kamera, cilj je matematički (a ne fizički) poravnati dvije kamere u jednu ravninu, što se opisuje u sljedećem potpoglavlju.

5.2. Epipolarna geometrija

Epipolarna geometrija (engl. epipolar geometry) je unutarnja projekcijska geometrija između dva pogleda koja ne ovisi o strukturi prizora već samo o unutarnjim parametrima kamera te njihovoj relativnoj poziciji. Slika 34. prikazuje shemu epipolarnih geometrija. Za svaku kameru sada postoji zasebni centar projekcije O_l i O_r , te pripadajuće projekcijske ravnine π_l i π_r . Točka Q iz svijeta ima projekcije q_l i q_r . Epipol (e_l) je definiran kao slika centra projekcije druge kamere Q_r . Ravnina koju formiraju točke Q , e_l i e_r naziva se epipolarna ravnina, a linije q_le_l i q_re_r epipolarnе linije.



Slika 35. Epipolarna geometrija

Točka Q koju vidimo projiciranu na desnu (ili lijevu) projekcijsku ravninu može se u stvarnosti nalaziti bilo gdje uzduž linije koja izlazi iz točke O_r i prolazi kroz q_r , s obzirom da s jednom kamerom nije moguće odrediti udaljenost do točke. Projekcija te linije na lijevu ravninu zapravo je epipolarna linija q_{le} . Drugim riječima, projekcija svih mogućih lokacija točke jedne slike je linija koja prolazi kroz korespondentnu i epipolarnu točku druge slike. Za bilo koju točku jedne slike vrijedi da njena korespondentna točka u drugoj slici leži na epipolarnoj liniji. Taj je uvjet poznat pod nazivom epipolarno ograničenje (engl. epipolar constraint). To pak znači da je umjesto dvodimenzionalne pretrage za korespondencijama dovoljna jednodimenzionalna pretraga, čime se znatno smanjenje složenost, a i omogućava odbacivanje velikog broja lažnih korespondencija.

5.3. Esencijalna i fundamentalna matrica

Kako bismo mogli pronaći epipolarne linije, potrebno je uvesti pojmove esencijalne i fundamentalne matrice. Esencijalna matrica E sadrži informacije o translaciji i rotaciji koje vežu dvije kamere u prostoru, odnosno daje nam vezu između točaka q_l i q_r u normiranom koordinatnom sustavu slike (u kojem je žarišna duljina $f = 1$). Fundamentalna matrica F sadrži, uz informacije koje sadrži esencijalna matrica, i intrinzične parametre obje kamere. Ona veže točke iz lijeve i desne slike q_l i q_r .

Uzmimo za referentnu točku O_l . Lokacija te točke u slici je točka Q_l , a ishodište druge kamere nalazi se u točki T . Točka Q u desnoj kameri jest $Q_r=R(Q_l - T)$.

Esencijalna matrica sadrži sve informacije o geometriji jedne kamere u odnosu na drugu, no ne sadrži nikakve informacije o samim kamerama. Međutim, u praksi, zanimaju nas koordinate u pikselima slike. Kako bismo mogli pronaći vezu između piksela jedne slike sa odgovarajućom epipolarnom linijom u drugoj slici, moramo iskoristiti intrinzične parametre kamera. Zato točku \mathbf{q} (vektor) supstituiramo točkom q i matricom intrinzičnih parametara kamere K , točnije $q=K\mathbf{q}$, odnosno $\mathbf{q}=K^{-1}q$.

Zaključak

Cilj ovog diplomskog rada bilo je razvijane algoritma koji će moći detektirati pravilnu strukturu kaveza za uzgoj ribe. Također je bilo potrebno istražiti mogućnost računanja dubine slike, odnosno kolika je udaljenost kamere od detektiranog objekta.

Ostvareni sustave se dijeli na tri glavne cjeline:

- Učitavanje slike
- Obrane slike
- Detekciju objekta

Programska implementacija se ostvarila korištenjem OpenCv biblioteke. Rezultati detekcije su se pokazali uspješnim, ako su uvjeti u kojim su nastale slike odnosno video zapisi bili relativno povoljni. Posebno se odnosi na kontrast, koji najviše utječe na detekciju, što se pokazalo u ovom radu. Međutim čim se malo promjene uvjeti uspješnost detekcije znatno opada, pa počinje detektirati i druge objekte u slici, koji nisu interesantni, odnosno pogrešne detekcije, a ti su se problemi otklonili korištenjem uvjeta konveksnosti zatvorenih kontura. Također rezultat detekcije uvelike ovisi i o prisutnosti drugih objekata u blizini objekta željenog detektirati.

Mogući daljnji koraci u vidu poboljšanja detekcije je korištenje histograma boja. Odnosno da se vrši detekcija na temelju boje koje se najviše pojavljuju u slici. Na taj način bi se moglo smanjiti broj krivih detekcija. Eventualni problem kod takve metode detekcije je smanjen spektar boja u vodi, te ovisi i o kvaliteti snimke koju radi kamera. U slučaju lošijih slika boja kaveza i drugih okolnih predmeta npr., algi je skoro identična, što se može lako vidjeti u obrađenom videu.

Literatura

- [1] IZVORNI KOD OPENCV BIBLIOTEKE, <http://opencv.org/downloads.html>
- [2] SISTEMA DE COLOR RGB, 30. travnja 2010,
<http://www.laguerradelosmundos.net/sistema-de-color-rgb/,10.6.2013.>
- [3] TORNADO CHARTS AND DOT PLOTS, *Jon Peltier*, 2. ožujak 2008.,
<http://peltiertech.com/WordPress/tornado-charts-and-dot-plots/>, 10.6.2013.
- [4] IMAGE MOSAIC, *Benedict Brown, Philip Shilane*,
<http://www.cs.princeton.edu/~pshilane/class/mosaic/,15.6.2013.>
- [5] IMAGE MOSAIC, *Benedict Brown, Philip Shilane*,
<http://www.cs.princeton.edu/~pshilane/class/mosaic/,15.6.2013.>
- [6] SATOSHI SUZUKI, Computer Vision, Graphics, and Image Processing, *Topological structural analysis of digitized binary images by border following*, 1985.
- [7] SATOSHI SUZUKI, Computer Vision, Graphics, and Image Processing, *Topological structural analysis of digitized binary images by border following*, 16. prosinac 1983., <http://wenku.baidu.com/view/6cb52ede360cba1aa811dad5.html>, 15.6.2013.
- [8] GREEN'S THEOREM, Wikipedia, http://en.wikipedia.org/wiki/Green%27s_theorem, 20.6.2013.
- [9] GARY BRADSKI, ADRIAN KAEBLER, *Learning opencv computer vision with the opencv library*, United States of America, 2008.
- [10] RICHARD HARTLEY, ANDREW ZISSEMAN, *Multiple View Geometry in computer vision*, Cambridge UK, 2003.
- [11] OPENCV TUTORIALS, <http://docs.opencv.org/doc/tutorials/tutorials.html>, 3.4.2013.
- [12] OPENCV TUTORIALS C++, *Shape Detection & Tracking using Contours*, Shermal Fernando, <http://opencv-srf.blogspot.com/2011/09/object-detection-tracking-using-contours.html>, 3.4.2013.
- [13] ADAPTIVE THRESHOLD COMPUTER VISION, Christoffer Baadsgaard, Søren Eskildsen, Kasper Rodil, Surenthan Sakthivel,
<http://www.cvmt.dk/education/teaching/f10/MED8/CV/Stud/832.pdf>, 10.6.2013.
- [14] MORPHOLOGICAL IMAGE PROCESSING,
<http://www.cs.auckland.ac.nz/courses/compsci773s1c/lectures/ImageProcessing-html/topic4.htm>, 10.6.2013.
- [15] ALAN SAMODOL, *Estimiranje strukture i gibanje stereo parom kamera*, Diplomski rad, Sveučilište u Zagrebu, 2012

Sažetak

PREPOZNAVANJE STRUKTURE KAVEZA ZA UZGOJ RIBA U VIDEO SLICI

Zadatak ovog rada bio je prepoznavanje pravilne strukture kaveza. Opisan je postupak instalacije OpenCv biblioteke. Opisane su tehnike koje su se koristile kod razvoja aplikacije za prepoznavanje pravilne strukture. Shema ostvarenog sustava sastoji se od nekoliko koraka: učitavanje slike, obrade slike te detekcija objekta. Također su prikazani eksperimentalni rezultati, koji su se pokazali koliko-toliko uspešnim. Istražena je mogućnost izračuna dubine slike.

Ključne riječi: OpenCv, adaptivni prag, glađenje, konture, pravilna struktura, stereo vizija

Summary

STRUCTURE RECOGNITION OF FISH CAGES IN A VIDEO IMAGE

The aim of this study was to identify the correct structure of the cage. The procedure to install OpenCV library is described. The methods that were used in the development of applications to identify the correct structure are described. Scheme of resulting system consists of several steps: upload images, image processing and object detection. It also shows the experimental results, which proved to be more or less successful. Explored the possibility of calculating the depth of the image.

Keywords: OpenCV, an adaptive threshold, smoothing, contour, regular structure, stereo vision

Skraćenice

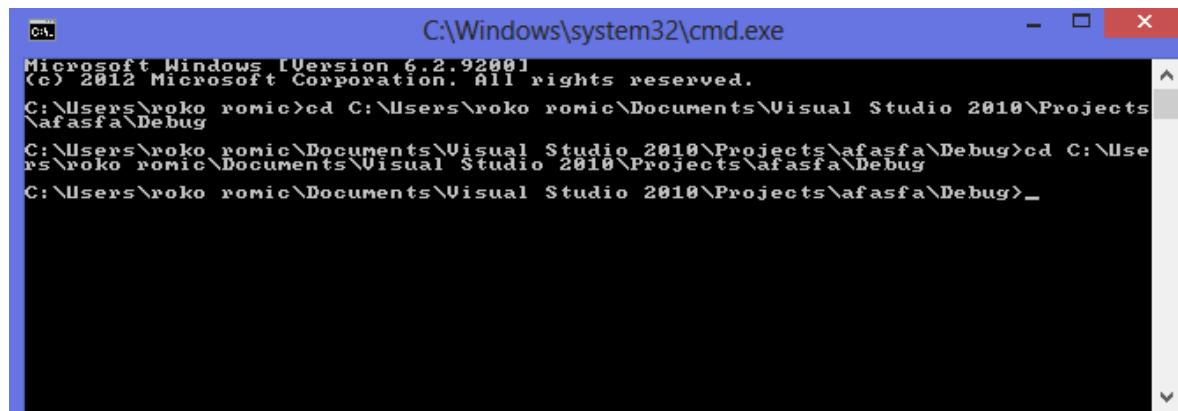
ROV	<i>Remotely operated underwater vehicle</i>	daljinsko upravljanje podvodno vozilo
RBG	<i>Red Blue Green</i>	crvena plava zelena
FPS	<i>Frame per seconds</i>	broj sličica u sekundi
CMD	<i>Command Prompt</i>	naredbeni prozor

Privitak

Upute za korištenje programske podrške

Da bi se moglo uspješno provesti testiranje aplikacije za video obradu, potrebno je slijediti iduće korake.

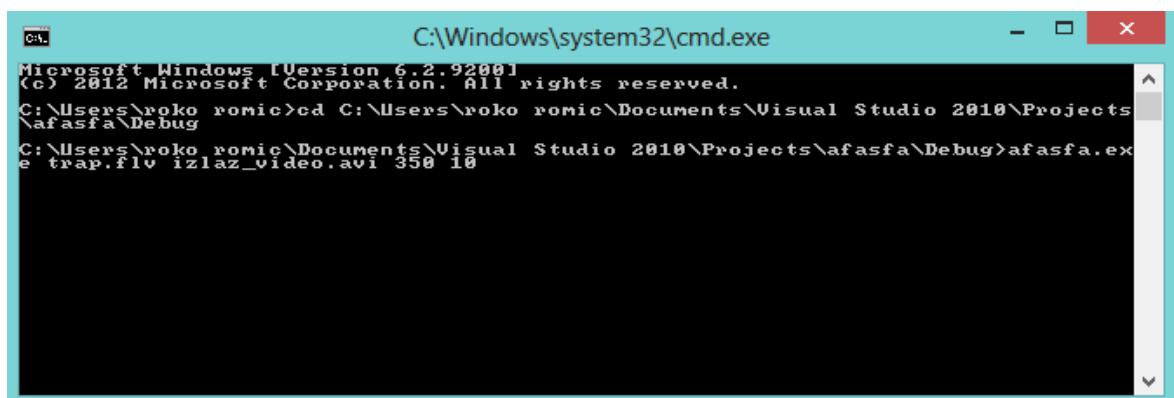
Najprije je potrebno postaviti putanju u komandom prozoru (eng. cmd) prema odredišnoj datoteci, u kojoj se nalazi *ime_programa.exe* te video materijali. Primjer je prikazan na idućoj slici.



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.2.9200]
(c) 2012 Microsoft Corporation. All rights reserved.
C:\Users\roko_romic>cd C:\Users\roko_romic\Documents\Visual Studio 2010\Projects\afasfa\Debug
C:\Users\roko_romic\Documents\Visual Studio 2010\Projects\afasfa\Debug>cd C:\Users\roko_romic\Documents\Visual Studio 2010\Projects\afasfa\Debug
C:\Users\roko_romic\Documents\Visual Studio 2010\Projects\afasfa\Debug>_
```

Slika 36. Primjer pozicioniranja na odgovarajuću putanju

Daljnji korak je, pravilno pozivanje aplikacije iz komandnog prozora. Pravilan poziv je slijedećeg oblika <*ime_programa.exe, ime_ulaznog_videa, ime_izlaznog_videa, donja granica površine kontura, preskakanje određenih sličica*>. Primjer pravilnog pozivanja prikazan je na slijedećoj slici.



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.2.9200]
(c) 2012 Microsoft Corporation. All rights reserved.
C:\Users\roko_romic>cd C:\Users\roko_romic\Documents\Visual Studio 2010\Projects\afasfa\Debug
C:\Users\roko_romic\Documents\Visual Studio 2010\Projects\afasfa\Debug>afasfa.exe trap.flv izlaz_video.avi 350 10
```

Slika 37. Primjer pravilnog pokretanja aplikacije