Aplikacija za pronalaženje objekata u slici

Šušak, Andrea

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

APLIKACIJA ZA PRONALAŽENJE OBJEKATA U SLICI

Završni rad

Andrea Šušak

Osijek, 2018.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 09.07.2018.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada

Ime i prezime studenta:	Andrea Šušak
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R3835, 19.09.2017.
OIB studenta:	97755242831
Mentor:	Doc.dr.sc. Mirko Köhler
Sumentor:	
Sumentor iz tvrtke:	
Naslov završnog rada:	Aplikacija za pronalaženje objekata u slici
Znanstvena grana rada:	**Obradba informacija (zn. polje računarstvo)**
Predložena ocjena završnog rada:	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 2 bod/boda Jasnoća pismenog izražavanja: 2 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	09.07.2018.
Datum potvrde ocjene Odbora:	16.07.2018.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 27.08.2018.

Ime i prezime studenta:	Andrea Šušak
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R3835, 19.09.2017.
Ephorus podudaranje [%]:	12

Ovom izjavom izjavljujem da je rad pod nazivom**: Aplikacija za pronalaženje objekata u slici**

izrađen pod vodstvom mentora Doc.dr.sc. Mirko Köhler

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD

Slikom se, u generalnom smislu, mogu smatrati svi mediji koje ljudsko oko može vidjeti, kao što su nepokretna slika, video, animacija, grafika, crteži, grafikoni, pa čak i tekst.[1] S obzirom da čovjek većinu informacija iz stvarnog svijeta dobiva vizualnim putem, nastaje sve veća potreba za podrobnijom analizom slike kako bi se iz nje izvuklo što više korisnih informacija. U svrhu mogućnosti elektroničke percepcije ljudskog vida i digitalnog razumijevanja slike, razvija se područje računalnog vida. Računalni vid predstavlja interdisciplinarno područje koje obuhvaća metode obrade, analiziranja i razumijevanja slike ili slijeda slika u svrhu dobivanja simboličkih i numeričkih informacija te temeljne tehnologije za automatsku analizu slike. Pri analizi slike nas često neće zanimati slika u cjelini, već samo određeni dijelovi slike ili objekti koji se na njoj nalaze. Kako bi uštedili na vremenu i smanjili količinu podataka koji će se analizirati određenim metodama računalnog vida, potrebno je provesti segmentaciju slike. Svrha segmentacije je podijeliti sliku na segmente sličnih svojstava kako bi se razdvojili objekti od interesa i pozadina slike, pa tako i olakšala daljnja analiza.

Cilj rada je realizirati aplikaciju za pronalaženje objekata na slikama na temelju njihove boje. Pri tome ćemo se koristiti jednom od metoda segmentacije slike-metoda zasnovana na izrastanju područja (engl. *Region growing segmentation*). Prema ovoj metodi, pojedine točke slike, odnosno pikseli, formiraju područje, odnosno regiju prema predefiniranom kriteriju sličnosti. Segmentacija će se primjeniti u RGB modelu boja pa će tako kriterij sličnosti biti Euklidska udaljenost između RGB vektora.

U drugom poglavlju ćemo se upoznati s osnovnim konceptima i metodama segmentacije slike. Razvoj aplikacije je prikazan u trećem poglavlju, a njeno testiranje i rezultati prikazani su u četvrtom poglavlju ovog rada.

1.1. Zadatak rada

Zadatak rada je napraviti aplikaciju u C# za pronalaženje objekata na slikama. Potrebno je objasniti poznate metode za pronalaženje objekata i implementirati najbolju u aplikaciji.

2. SEGMENTACIJA SLIKE

Segmentacija slike je proces podjele slike na područja ili kategorije koje odgovaraju različitim objektima ili dijelovima objekata. Svaki piksel u slici je lociran u jednoj od više kategorija. Segmentacija se smatra uspješnom ukoliko zadovoljava sljededeće kriterije:

1. Područja moraju biti uniformna i homogena s obzirom na određenu karakteristiku
2. Njihova unutrašnjost mora biti jednostavna, bez puno praznina
3. Susjedna područja moraju imati značajno različite vrijednosti s obzirom na uniformne karakteristike
4. Granice svakog segmenta moraju biti što jednostavnije, glatke te prostorno točne.[2]

Postoje tri glavna pristupa segmentaciji: metoda segmentacije temeljena na određivanju praga (engl. *thresholding*), metoda segmentacije temeljena na rubovima (engl. *edge-based*) i metoda segmentacije temeljena na regijama (engl. *region-based*).

2.1. Metoda segmentacije temeljena na određivanju praga

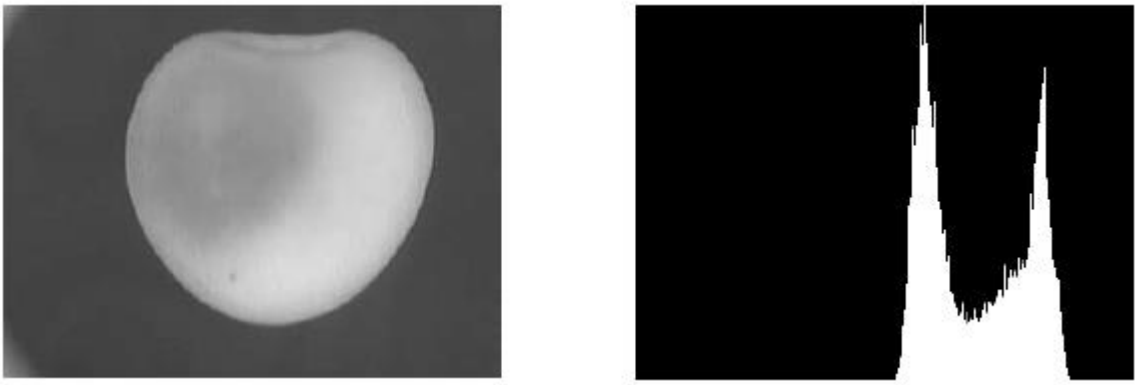
Metoda segmentacije temeljena na određivanju praga je najjednostavnija i najčešće korištena metoda segmentacije. Na sivim slikama (engl. *Grayscale image*) razdvaja objekte od pozadine na temelju intenziteta osvijetljenosti. Na temelju raspodjele sivih tonova u danoj slici, određena vrijednost osvijetljenosti se izabire kao prag T koji odvaja točke koje pripadaju objektu od značaja od onih koje pripadaju pozadini. Za zadanu vrijednost praga T, proces segmentacije odvija se na način:

$$g(x,y) = \begin{cases} 1, & f(x,y) \leq T \\ 0, & f(x,y) > T \end{cases} \tag{2-1}$$

Gdje $g(x,y) = 1$ predstavlja objekte, a $g(x,y) = 0$ pozadinu ili obrnuto.[3] Ovakva metoda segmentacije je pogodna za slike visokog kontrasta na kojima se objekti jasno razlikuju od pozadine i međusobno se ne dodiruju. Uspješnost ovisi o odabiru praga te vrlo često jedan prag nije dovoljan za cijelu sliku zbog varijacija uzrokovanih nejednakim osvijetljenjem i drugim faktorima. Prag može biti određen isprobavanjem više pragova, kako bi našli onaj koji najbolje odgovara ili automatski, analizom histograma.

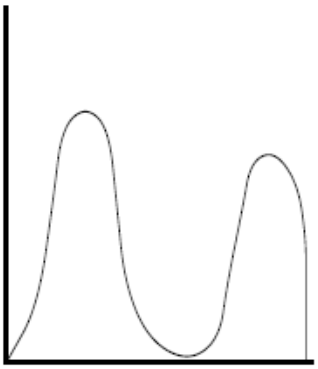
2.1.1. Analiza histograma

Histogram je grafički prikaz raspodjele tonova. Ako govorimo o sivoj skali fotografije(engl. *Grayscale image*), raspodjela tonova se vrši u tri područja na horizontalnoj osi histograma. Najtamnija nijansa označena je sa 0 a najsvjetlija sa 255.

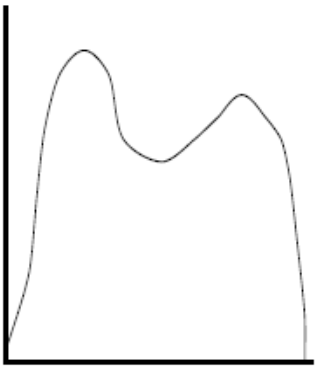


Slika 2.1. Fotografija višnje s pripadajućim histogramom (slika preuzeta iz [4])

Pomoću histograma određujemo prag koji binarizira sliku, odnosno dijeli piksele na tamne i svijetle (pozadina i objekt od interesa). Za automatski odabir praga potrebno je detektirati minimume i maksimume histograma. Ukoliko histogram ima istaknute maksimume između kojih je minimum, prag se lako određuje kao vrijednost srednjeg minimuma histograma. Problemi nastaju ukoliko dolazi do preklapanja minimuma i maksimuma,kao što je prikazano na slici 1.1.b), kada postaje teže odrediti zadovoljavajući prag.



Slika 2.2.a) Jasno izraženi maksimumi



Slika 2.2.b) preklapanje (preuzete iz [4])

2.1.2. Otsu metoda

Otsu metoda je najčešće korištena metoda automatskog pronalaženja praga. Ova metoda izračunava optimalan prag segmentacije tako da vrijednost varijance unutar segmenta bude minimalna, a vrijednost varijance između različitih segmenata bude maksimalna. Jednostavnije objašnjeno, intenziteti piksela koji pripadaju istom segmentu moraju biti blizu jedan drugom, dok intenziteti piksela koji ne pripadaju istom segmentu moraju biti daleko jedni od drugih.

2.2. Metoda segmentacije temeljena na pronalasku rubova

Rubovi na slikama predstavljaju nagle promjene u intenzitetu susjednih točaka (piksela). Obično se nalaze na granicama između objekata. Metoda segmentacije temeljena na rubovima je najpoznatiji pristup za otkrivanje spomenutih diskontinuiteta. Pronalazak ruba znatno smanjuje količinu podataka koji se trebaju obraditi kako bi se, u konačnici, objekti odvojili od ostatka slike. Najčešći problemi kod ove metode segmentacije su detekcije rubova tamo gdje oni ne postoje ili slučaj kada se rub ne detektira tamo gdje postoji. To može biti uzrokovano šumovima, koji su na slikama također predstavljeni naglom promjenom intenziteta, lošom kvalitetom slike, isprekidanim rubom i sl. Kako bi se riješili neželjenih šumova, koji se uglavnom sastoje od visokih frekvencija, na slici moramo primjeniti niskopropusni filtar. Jedan od njih je Gaussov filtar koji vrijednost svakog piksela određuje na temelju susjednih piksela, odnosno, svaki piksel čiji intenzitet odskače od susjednih, ovaj filtar stapa s okolinom.

Gausova 1-D distribucija:

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

Gdje je σ standardna devijacija Gausove razdiobe a x udaljenost od srednjeg piksela.

2.2.1. Cannyjev detektor ruba

Rubovi se mogu detektirati uz pomoć raznih detektora, a najčešće korišten je Canny detektor. Ovaj detektor je dizajniran tako da je lažno detektiranje ruba svedeno na minimum, svaki detektiran rub je što bliže stvarnom rubu te se svaki rub detektira samo jednom. Koraci segmentacije su sljedeći:

1. Zaglađivanje slike Gaussovim filtrom kako bi eliminirali šumove

2. Derivacija zaglađene slike kako bi se odredio gradijent i kut gradijenta
3. Stanjivanje rubova pomoću vrijednosti gradijenta i njegova kuta
4. Uspoređivanje točaka s gornjim i donjim pragom kako bi odredili pripadaju li zaista rubovima. Točke čije su vrijednosti veće od gornjeg praga pripadaju rubu, dok se točke s vrijednostima manjim od praga odbacuju. Točke čije se vrijednosti nalaze između gornjeg i donjeg praga pripadaju rubu ukoliko su povezane s nekom od rubnih točaka.

Možemo zaključiti da je metoda segmentacije temeljena na pronalasku rubova osjetljiva na šumove te se, kao samostalna, ne treba koristiti za segmentaciju slika niskog kontrasta.

Slika 2.3. Rezultat Canny detektora ruba (slika preuzeta iz [5])

2.3. Metoda segmentacije temeljena na regijama

Cilj segmentacije bazirane na rubovima je razdvojiti sliku na područja, odnosno regije, tako da regiju formiraju točke sličnih svojstava. Dok su prethodno objašnjene metode određivale granice između objekata od interesa i pozadine, ova metoda segmentacije direktno pronalazi regije. Pri tome moraju biti zadovoljeni sljedeći uvjeti:

1. Svaka točka mora pripadati nekoj od regija
2. Točke jedne regije moraju biti povezane prema predefiniranom kriteriju
3. Različite regije moraju biti odvojene
4. Točke koje pripadaju istoj regiji moraju imati slična svojstva.[6]

Postoje dva pristupa ove segmentacije, to su metoda izrastanja područja i metoda spajanja i razdvajanja.

2.3.1. Metoda spajanja i razdvajanja

U ovoj metodi početni segment predstavlja cijela slika. Ukoliko regija nije uniformna, segmenti se razdvajaju u četiri podregije, a ukoliko postoje četiri uniformne podregije, one se spajaju u jednu regiju. Uniformnost se može mjeriti razlikama između najtamnije i najsvjetlije točke.[2]

2.3.2. Metoda izrastanja područja

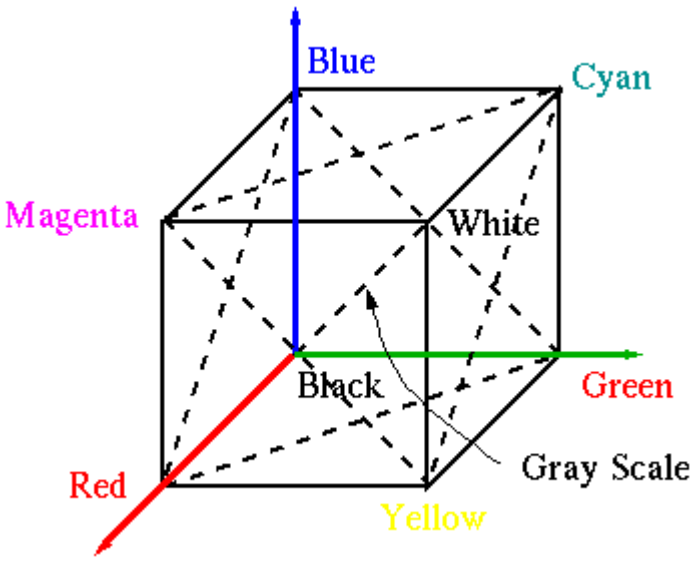
Metodom izrastanja područja se pojedine točke ili podregije grupiraju u veće regije ovisno o predefiniranom kriteriju. Segmentacija počinje odabirom jedne ili više početnih točaka, odnosno piksela (engl. *seed pixel*). Ovisno o tome što uspoređujemo, postoje tri tipa metode izrastanja područja:

a) **Metoda pomoću sličnosti točaka** (engl. *single linkage region growing*)
   Svaki piksel predstavlja čvor grafa. Susjedni pikseli, čija su svojstva dovoljno slična, se povezuju granom. Na taj način pojedini segmenti predstavljaju maksimalni skup točaka koje pripadaju jednom povezanom skupu.
b) **Metoda pomoću sličnosti okolina** (engl. *hybrid linkage region growing*)
   Svakom pikselu se dodjeljuje svojstveni vektor koj ovisi o KxK susjedstvu tog piksela. Neke od metoda ove segmentacije koriste maske za detekciju ruba, kako bi se utvrdilo pripadaju li pikseli rubu. Problemi se mogu javiti zbog prekinutosti rubova, zbog čega je izbor operatora za pronalaženje ruba ključan za uspješnost ove metode.
c) **Metode koje ispituju sličnost točke i regije** (engl. *centroid linkage region growing*)
   U ovoj metodi pikseli nisu združeni zbog međusobnih sličnosti, kao što je to slučaj u prethodnim metodama. Slikom se prolazi prema predefiniranom redoslijedu te se vrijednost pojedine točke uspoređuje sa srednjom vrijednošću već klasificiranih točaka regije, koja ne mora nužno biti kompletna. Ukoliko je vrijednost promatrane točke slična srednjoj vrijednosti prethodno klasificiranih točaka, ta točka se dodaje regiji. Odstupanja se mogu javiti ovisno o redoslijedu prolaska slikom.

Kriterij za sličnost može biti temeljen na bilo kojoj od karakteristika slike, kao što su srednji intenzitet, varijanca, boja, tekstura itd, a sličnost se može mjeriti Euklidskom udaljenošću. Izbor početne točke je ključan u ovoj metodi i ovisi o prirodi problema.

2.4. Segmentacija u RGB modelu boja

U RGB modelu boja, svaka boja se sastoji od primarnih komponenti crvene (engl. ***R**ed*), zelene (engl. ***G**reen*) i plave boje(engl. ***B**lue*). Slike prikazane u ovom modelu boja se sastoje od tri komponente, po jedna za svaku od primarnih boja. Model se temelji na Kartezijevom koordinatnom sustavu. Na slici 2.4. možemo vidjeti raspored boja ovog modela.



Slika 2.4. RGB model boja (slika preuzeta iz [7])

Crvena, zelena i plava boja smještene su na osima, crna boja se nalazi u ishodištu, dok je bijela najudaljenija od ishodišta. RGB model predstavlja aditivni model boja jer su sekundarne boje kreirane dodavanjem crvene, zelene i plave na crnu(0, 0, 0). Različite boje u ovom modelu su točke na kocki ili unutar nje te su definirane vektorom koji se pruža od ishodišta. Siva fotografija nastaje kada su sve tri komponente (crvena, zelena, plava) izjednačene. Broj bitova potrebnih za prikaz boje svakog piksela (engl. *bits per pixels*) u RGB modelu se naziva dubina boje (engl. *colour depth*). Većina formata za spremanje slika, kao i većina prikaza na zaslonu računala i mobilnih telefona, koriste 24 bita za prikaz svakog piksela, odnosno po 8 bitova za prikaz svake od tri komponente. Gotovo svi slučajevi u kojima postoji 32 bita po pikselu znače da se 24 koriste za boju, a preostalih 8 je alfa kanal korišten za transparentnost ili neiskorišten.

Iako RGB model boja odgovara činjenici da je ljudsko oko snažno perceptivno na crvenu, zelenu i plavu boju, ovaj i slični modeli boja nisu primjereni za opisivanje boja u smislu ljudske interpretacije istih. Na primjer, boju određenog predmeta ne opisujemo u postotcima primarnih boja koje ju stvaraju već ju opisujemo po njenoj nijansi i svjetlini. No, hardverska implementacija

i segmentacija slike su područja gdje korištenje RGB modela daje bolje rezultate u odnosu na druge modele čiji su opisi bliži ljudskoj interpretaciji.

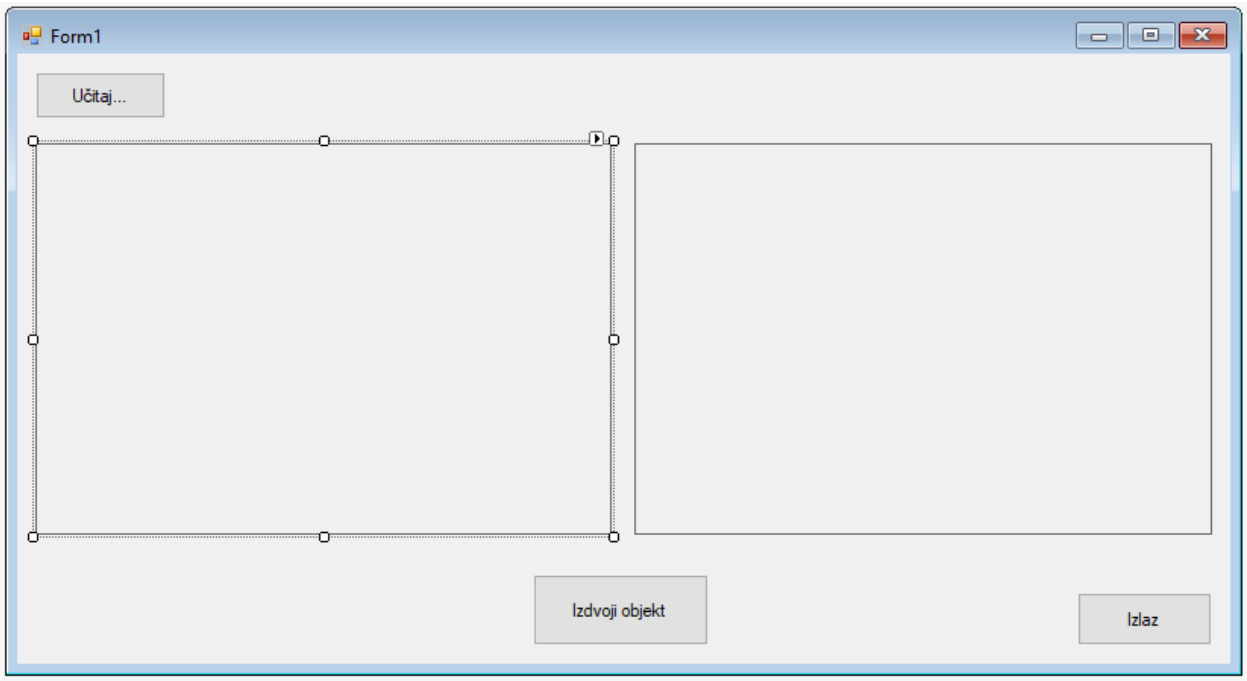
Recimo da segmentacijom želimo izdvojiti objekt specifične boje. S obzirom na skup točaka koje predstavljaju boju od interesa, dobivamo procjenu prosječne boje koju želimo izdvojiti. Tu boju definiramo vektorom a. Zadaća segmentacije je provjeriti svaki piksel i definirati pripada li njegova boja zadanom rasponu ili ne. Za mjeru sličnosti boja najčešće se koristi Euklidska udaljenost. Promatrani vektor z će biti sličan zadanom ukoliko je udaljenost između njih manja od zadanog praga D_0:

$$D_{z,a} = \sqrt{(z_R - a_R)^2 + (z_G - a_G)^2 + (z_B - a_b)^2} \tag{2-3}$$

Pri tome, indeksi R, G i B označavaju RGB komponente vektora a i z. [6] Točke koje zadovoljavaju uvjet $D_{z,a} \leq D_0$ zadovoljavaju predefiniran kriterij za segmentaciju boje, dok točke za koje vrijedi $D_{z,a} > D_0$ se nalaze izvan definiranog raspona boje i ne čine dio objekta koji želimo izdvojiti.

3. RJEŠENJE ZADATKA I KORIŠTENI ALATI

Aplikacija je realizirana u *Microsoft Visual Studio 2017* okruženju, jezik koji je korišten je C# , a s obzirom da aplikacija treba pronalaziti objekte učitane slike, korišten je Aforge.NET okvir. Na slici 3.1. je prikazana Windows Forma s pripadajućim gumbima i prozorima.



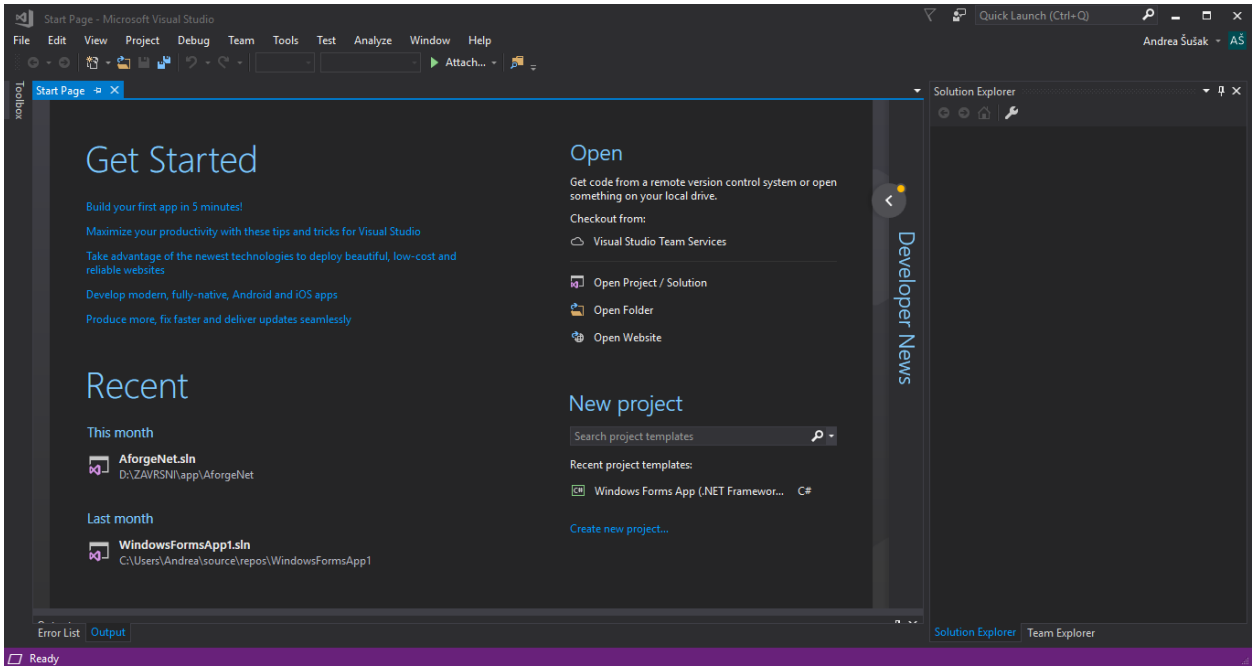
Slika 3.1. Aplikacija za pronalaženje objekata u slici

Forma se sastoji od gumba za učitavanje slike te dva prozora, lijevi za prikaz originalne slike, a desni za prikaz slike nakon postupka segmentacije. Kada učitamo sliku, u oba prozora će biti prikazana originalna slika. Klikom na gumb *Izdvoji objekt* pozivaju se metode za segmentaciju slike i pronalazak objekata. U ovoj aplikaciji će biti izdvojeni objekti crvene boje.

3.1. Microsoft Visual Studio

Microsoft Visual Studio predstavlja integrirano razvojno okruženje. Pogodan je za razvoj računalnih programa, kao i web stranica, web aplikacija, web usluga te mobilnih aplikacija [8]. Podržava 36 različitih programskih jezika, neki od njih su C, C++, C#, Visual Basic, Java, HTML itd.

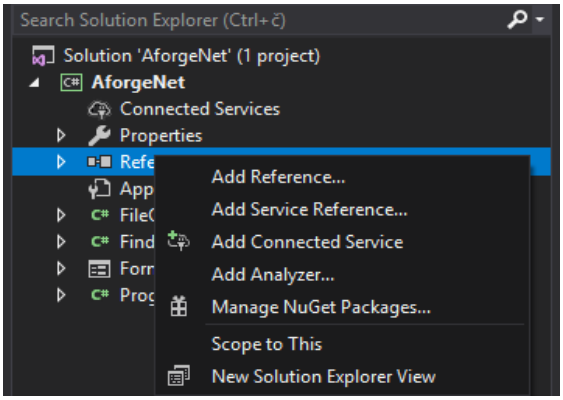
Microsoft Visual C# predstavlja Microsoftovu implementaciju programskog jezika C#. Zajedno s .NET okvirom i jezičnim uslugama podržava C# projekte. Jezične usluge, poput IntelliSense, su dio Visual Studia, dok je prevoditelj dostupan odvojeno, kao dio .NET okvira.



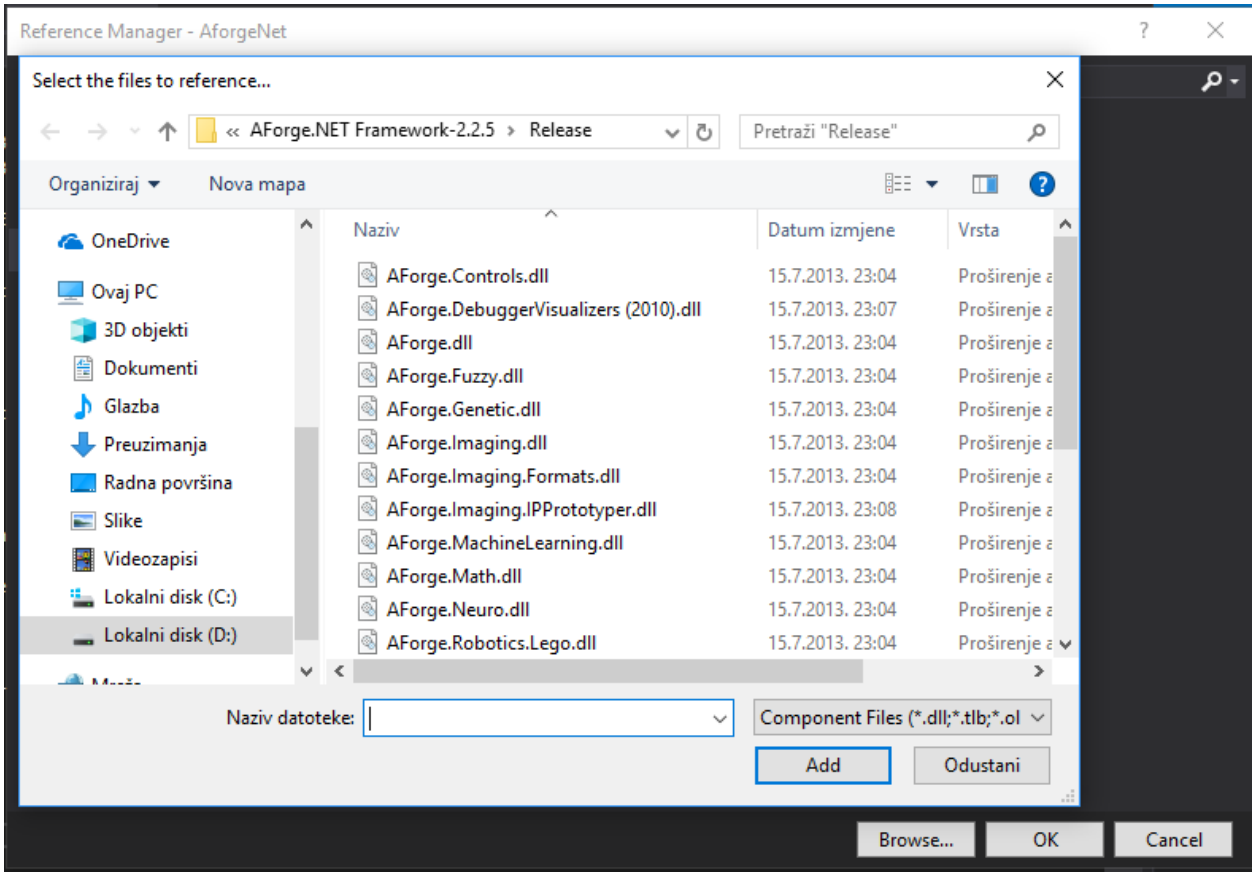
Slika 3.2. Sučelje Microsoft Visual Studia

3.2. AForge.NET

Aforge.Net je C# okvir otvorenog izvornog koda, dizajniran za razvoj i istraživanje na području računalnog vida i umjetne inteligencije. U spomenuto područje ubrajamo obradu slike, neuronske mreže, genetičke algoritme, robotiku i sl.[9]. Okvir se sastoji od brojnih biblioteka. Kako bi koristili biblioteke, potrebno je preuzeti instalacijsku datoteku sa sljedeće stranice: http://www.aforgenet.com/framework/downloads.html te ju raspakirati u jedan od direktorija na računalu. Nakon raspakiranja, u *Solution Exploreru* u Visual studiu potrebno je odrabrati *Add Reference*... te iz mape *Release*, koja se nalazi u preuzetoj datoteci, odabrati željene .dll datoteke. U ovom radu će se koristiti Aforge.Imaging biblioteka koja sadrži razne metode obrade slike kao što su različiti filteri, detektori itd.



Slika 3.3. Dodavanje referenci u Solution Exploreru



Slika 3.4. Dodavanje .dll datoteka

3.3. Aplikacija

U prilogu dokumenta se nalazi kompletan izvorni kod.

Realizacija aplikacije može se podijeliti u sljedeće cjeline:

1. Kreiranje bitmap datoteke
2. Primjenjivanje filtra kako bi se izdvojili objekti željene boje
3. Pronalaženje objekata na slici
4. Iscrtavanje granica između objekata.

Aplikacija se sastoji od tri glavne komponente : glavni program (*Form.cs*), klasa *FileOperations* te klasa *FindObject*. Unutar klase *FileOperations* definirana je metoda koja za učitanu sliku vraća datoteku bitmap formata (.bmp).

Bitmap u doslovnom prijevodu označava „mapu bitova“, odnosno predstavlja format u kojem je zapisana vrijednost RGB komponenti svakog pojedinog piksela učitane fotografije. Ovakav format podržavaju svi programi za obradu slike pa je zbog toga pogodan za izvršavanje ovog zadatka.

```
class FileOperations
{
    Bitmap NewImage;

    1 reference
    public Bitmap OpenFile()
    {
        OpenFileDialog ofd = new OpenFileDialog();
        ofd.InitialDirectory = "D:\\Fotografije";
        ofd.Filter = "images| *.jpg; *png; *.bmp";
        if(ofd.ShowDialog()==DialogResult.OK)
        {
            NewImage = new Bitmap(Image.FromFile(ofd.FileName));
        }
        return NewImage;
    }
}
```

Programski kod 3.1. FileOperations klasa

Nakon učitavanja, bitmap datoteka se kao parametar šalje metodi za segmentaciju unutar klase *FindObject*.

Za izdvajanje objekta željene boje najprije koristimo *EuclideanColorfiltering* klasu. Ova klasa sadrži filter koji objekt izdvaja od pozadine na temelju sličnosti RGB vrijednosti točaka. Kao mjera sličnosti se koristi Euklidska udaljenost koja je detaljnije opisana u potpoglavlju 2.4. Najprije se definira RGB vrijednost početnog piksela koja predstavlja središte sfere. U našem slučaju, to je RGB vektor s dominantnom crvenom komponentom, u kodu zapisano kao:

filter.CenterColor = new RGB(Color. FromArgb(215, 10, 30));

Nakon toga, potrebno je definirati maksimalnu udaljenost točaka koje će se smatrati sličnima prethodno definiranoj točki, odnosno, potrebno je definirati prag. Prag predstavlja radius sfere, u kodu zapisano kao:

filter.Radius = 110;

Ovime je definirano da će svaka točka koja je unutar sfere radiusa 110 (ili na njenoj površini) predstavljati dio detektiranog objekta, dok točke izvan sfere predstavljaju pozadinu. Točke koje pripadaju pozadini, poprimaju RGB vrijednosti (0, 0, 0) čime se smanjuje broj podataka koji se kasnije trebaju obrađivati na slici. Nakon definiranih parametara, metoda *ApplyInPlace(bmap);* kao argument prima bitmap datoteku te na njoj primjenjuje filter.

```
EuclideanColorFiltering filter = new EuclideanColorFiltering();
filter.CenterColor = new RGB(Color.FromArgb(215, 10, 30));
filter.Radius = 110;
filter.ApplyInPlace(bmap);
```

Programski kod 3.2. EuclideanColorFiltering klasa

Kako su objekti odvojeni na crnoj pozadini, potrebno ih je detektirati. Za to koristimo BlobCounter klasu koja je također dostupna unutar Aforge.NET okvira. BlobCounter klasa sadrži razne metode za izdvajanje objekata te njihovu manipulaciju. Akronim BLOB (engl. *Binary Large Object*) se prevodi kao „veliki binarni objekt“ i označava skup piksela sličnih vrijednosti intenziteta koje se razlikuju od pozadine.Klasa podržava obradu sivih slika dubine 8 bita po pikselu i slika u boji dubine 24 ili 32 bita po pikselu čija širina mora biti najmanje 2 piksela.

U programskom kodu 3.3. je prikazan dio koda s BlobCounter klasom i njenim metodama. Najprije su definirane dimenzije najmanjih objekata koji će se detektirati, u kodu prikazano kao:

bc.MinWidth = 5;
bc.MinHeight = 5;

U našem slučaju minimalna širina (*bc.MinWidth*) i minimalna dužina (*bc.MinHeight*) su postavljeni na 5, što znači da će najmanji detektirani objekt biti dimenzija 5x5 piksela čime se izbjegava detektiranje šumova kao objekata.

Metoda koja traži objekte je :

bc.ProcessImage(bmap);

Ova metoda kao argument prima prethodno filtriranu sliku.

```
BlobCounter bc = new BlobCounter();
bc.FilterBlobs = true;
bc.MinWidth = 5;
bc.MinHeight = 5;
bc.FilterBlobs = true;
bc.ProcessImage(bmap);
Rectangle[] rects = bc.GetObjectsRectangles();
foreach (Rectangle recs in rects)
    if (rects.Length > 0)
    {
        foreach (Rectangle objectRect in rects)
        {

            Graphics g = Graphics.FromImage(mImage);
            using (Pen pen = new Pen(Color.FromArgb(160, 255, 160), 5))
            {
                g.DrawRectangle(pen, objectRect);
            }

            g.Dispose();
        }

    }
```

Programski kod 3.3. BlobCounter klasa

Kako bi na originalnog slici iscrtali granice oko detektiranih objekata, potrebne su nam informacije o njihovom položaju.

Metoda *bc.GetObjectsRectangles();* vraća polje pravokutnika koji se nalaze na položaju detektiranih objekata. Svaki pravokutnik određen je svojom širinom, visinom i lijevim gornjim kutom koji predstavlja lokaciju.

Kako bi na originalnoj slici nacrtali pravokutnike, potrebna je instanca klase *Graphics* te instanca klase *Pen*. Instanca klase *Graphics* pruža metodu *DrawRectangle,* dok se instancom klase *Pen* definiraju širina linije i boja kojom se crta. Instanciranje klase *Gaphics* prikazano je linijom koda:

Graphics g = Graphics.FromImage(mImage);

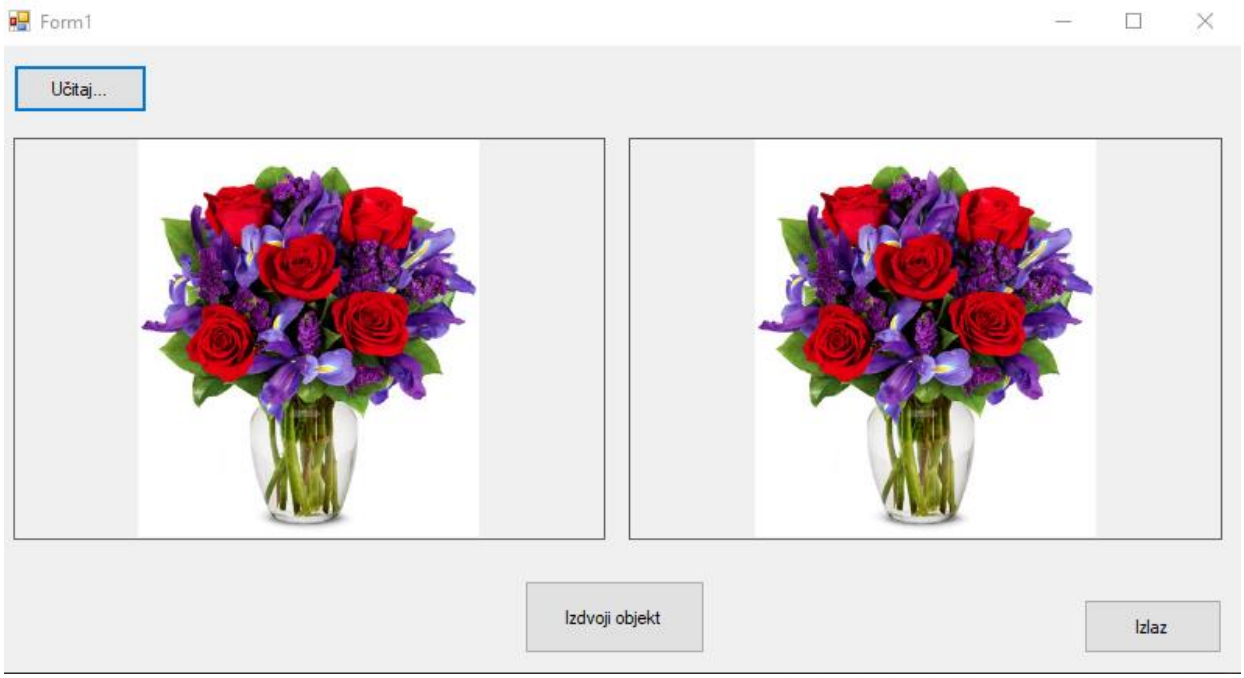
Pri tome se metodi *FromImage();* predaje originalna slika. Funkcija *DrawRectangle* prima dva argumenta, prvi je objekt za crtanje a drugi je struktura koja predstavlja pravokutnik koji će se crtati.

4. REZULTATI

Za testiranje aplikacije koristimo fotografiju dimenzija 340x397 piksela, te dubine 24 bita po pikselu.

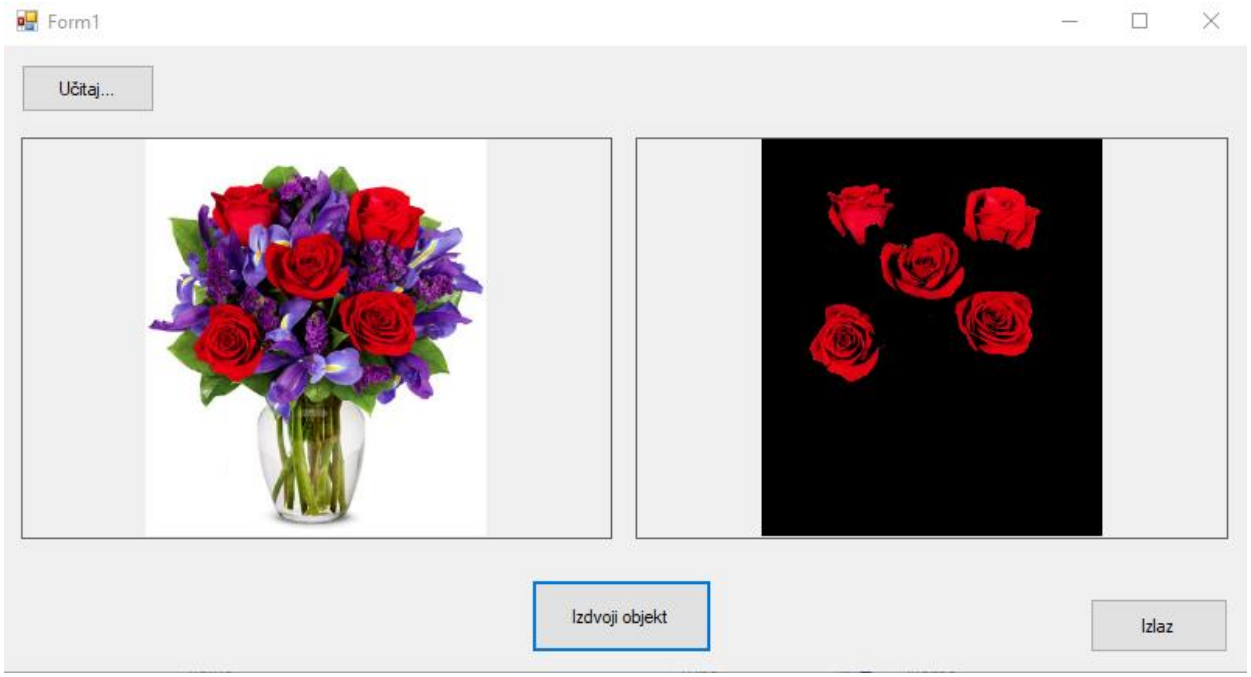
Slika 4.1. Originalna slika

Klikom na gumb *Učitaj* sa računala učitavamo sliku na kojoj ćemo detektirati crvene objekte. Na slici 4.2. je prikazano sučelje aplikacije nakon učitavanja slike.



Slika 4.2. Sučelje aplikacije

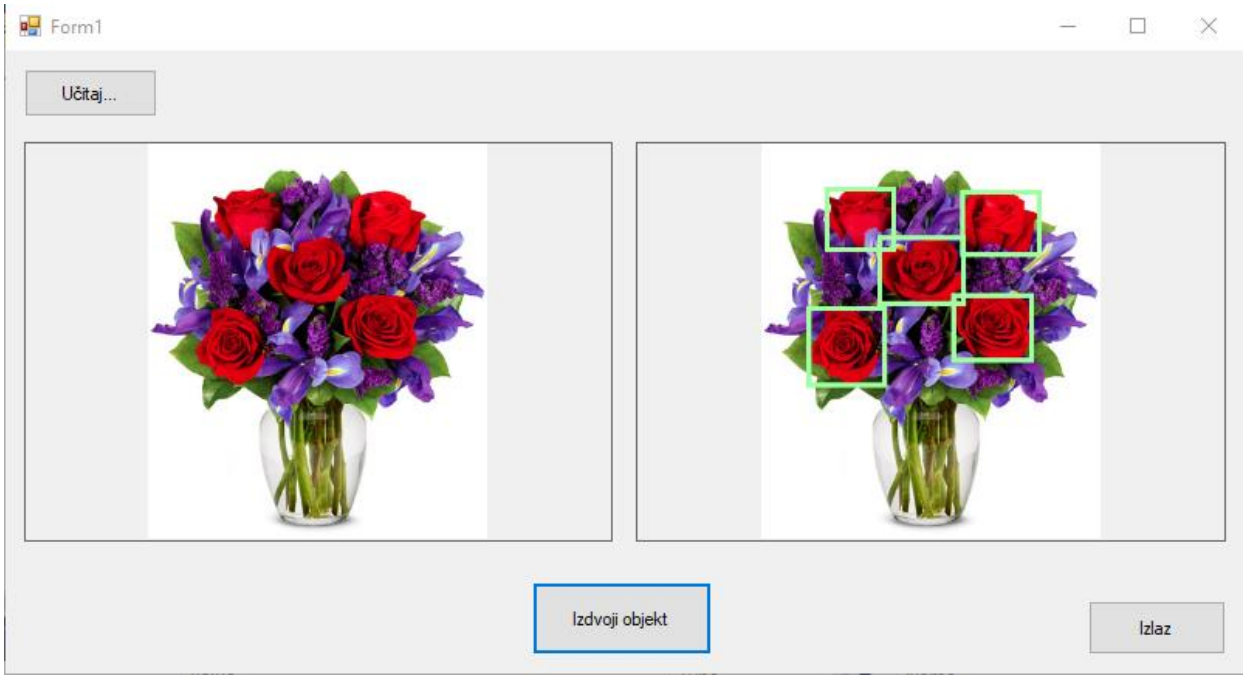
Kao što je opisano u prethodnom poglavlju, za izdvajanje objekata crvene boje koristimo *EuclideanColorFiltering* klasu. Na slici 4.3. je prikazan rezultat filtriranja.



Slika 4.3. Rezultat filtriranja pomoću Euklidske udaljenosti

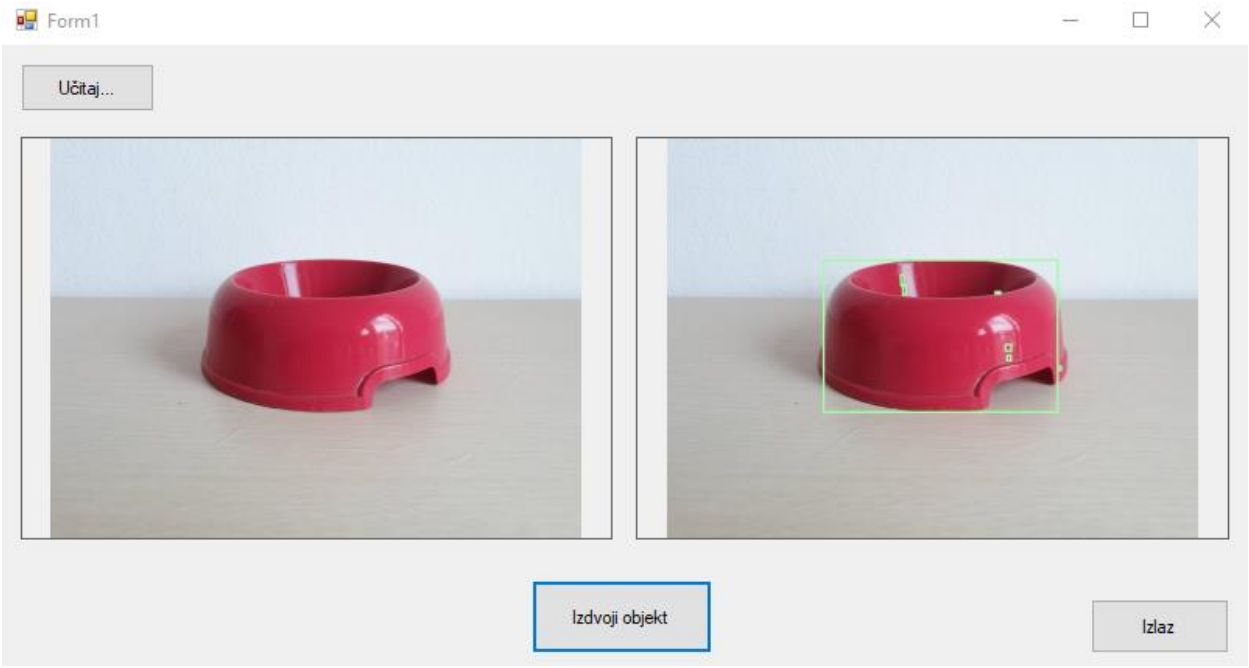
Filtrirani su svi značajni crveni objekti sa slike, pa je filtriranje uspješno. Potrebno je napomenuti kako kriterij euklidske udaljenosti ne uzima u obzir fizičku udaljenost piksela, već samo razliku

njihovih intenziteta. Vidimo kako su točke koje pripadaju pozadini poprimile crnu boju. Nakon filtriranja objekata željene boje, za pronalazak pojedinih objekata potrebno je koristiti *BlobCounter* klasu , te klasu *Graphics* za crtanje rubova oko objekata. Metode koje su korištene, kao i postavljeni parametri za crtanje linije prikazani su u prethodnom poglavlju. Krajnji rezultat prikazan je na slici 4.4.



Slika 4.4. Krajnji rezultat

Na slici 4.4. je vidljivo kako su detektirani svi željeni objekti. Međutim, potrebno je napomenuti kako se program testirao na fotografiji visoke rezolucije na kojoj nisu zastupljeni šumovi niti ima značajnih preklapanja između objekata koje želimo detektirati.



Slika 4.5. Krajnji rezultat

Na slici 4.5. vidi se da postoje manji segmenti na detektiranom objektu koji su pogrešno detektirani kao zasebni objekti. Ovakav rezultat je uzrokovan lošim osvijetljenjem i presijavanjem na objektu. Iako je rezultat lošiji nego na slici 4.4., i dalje je zadovoljavajuć jer je u konačnici detektiran objekt željene boje.

Slika 4.6. Krajnji rezultat

Slika 4.6. se sastoji od dva crvena jastuka i pseće ogrlice sa motivima crvenih kostiju. Na ovoj slici također se pojavljuju manji segmenti koji su pogrešno detektirani kao zasebni objekti te su dva jastuka detektirani kao jedinstveni objekt jer se među njima pojavljuje peklapanje.

5. ZAKLJUČAK

Zadatak ovog rada bio je objasniti poznate metode pronalaženja objekata u slici te najbolju implementirati u C# aplikaciji. U radu su, kroz nekoliko potpoglavlja, opisane metode segmentacije slike koje su polazište pri detekciji objekata. Realizirana aplikacija za detekciju objekata koristi metodu izrastanja područja čiji je kriterij za izdvajanje objekata Euklidska udaljenost RGB vrijednosti točaka slike. Aplikacija je testirana na nekoliko fotografija različite rezolucije. Testiranjem je utvrđeno da kvaliteta slike, pojava preklapanja između objekata, osvijetljenje, presijavanje i sl. značajno utječu na uspješnost detekcije objekata na slikama. Korištena metoda dala je najbolje rezultate na slici visoke rezolucije gdje se objekti od interesa jasno razlikuju od pozadine. Problemi se javljaju na slikama lošije kvalitete, gdje su prisutna preklapanja između objekata, loše osvijetljenje ili presijavanje pa se pojavljuju manji segmenti koji su na detektiranom objektu detektirani kao zasebni objekti ili se, zbog preklapanja, dva objekta iste boje detektiraju kao jedinstveni objekt. Navedene probleme možemo reducirati boljom obradom slike prije provođenja segmentacije i detekcije ili promjenom praga.

LITERATURA:

[1] Yu-Jin Zhang, *Advances in Image and Video Segmentation*, Tsinghua University, Beijing, China, 2006.

[2] R. M. Haralick, L. G. Shapiro, *Image segmentation Techniques*, Computer Vision, Graphics and Image processing, 29, 100-132, January 1985.

[3] H. H. A Kadouf, Y. M. Mustafah, Colour-based Object Detection and Tracking for Autonomous Quadrotor UAV, iop.science, dostupno na: http://iopscience.iop.org/article/10.1088/1757-899X/53/1/012086/pdf [22.05.2018.]

[4] L. G. Shapiro, G. Stockman, *Computer Vision*, Prentice Hall , Upper Saddle River, New Jersey, USA, 2001.

[5] T. Nguyen, Image Processing: *Canny Edge Detection in C++*, Joyful Engineering, 2016. dostupno na: https://turbosnu.wordpress.com/2016/01/12/image-processing-canny-edge-detection-in-c/ [25.05.2018.]

[6] R. C. Gonzales, R. E. Woods, *Digital Image Processing*, Prentice Hall, Upper Saddle River, New Jersey, USA, 1992.

[7] R. Wang, *Color Models*, RGB/YCM model, dostupno na: http://fourier.eng.hmc.edu/e161/lectures/color_processing/node1.html [25.05.2018.]

[8] Visual Studio IDE overwiev, dostupno na: https://docs.microsoft.com/en-us/visualstudio/ide/visual-studio-ide [29.06.2018.]

[9] AforgeNet Framework, dostupno na: http://www.aforgenet.com/framework/ [29.06.2018.]

SAŽETAK

U radu su objašnjene metode segmentacije slike, koje su ključan korak u obradi slike i detekciji objekata. Prikazana je C# aplikacija koja na učitanoj slici pronalazi i izdvaja objekte crvene boje pomoću metode izrastanja područja. Prikazana implementacija počinje s jednom točkom kao središtem regije te područje izrasta na temelju sličnosti piksela, a kao mjera sličnosti, koristi se Euklidska udaljenost RGB vektora.

Ključne riječi: C#, detekcija objekata, Euklidska udaljenost, segmentacija slike

ABSTRACT

An application for finding objects in the image

In this B.A. thesis, segmentation methods, which are an essential step towards image processing and object detection, were explained. A C# application, which searches and extracts red colored objects using a region growing method, was represented. Represented implementation starts with one point as a center of the region and then the region is grown based on a pixel similarity, and as a similarity criterion, Euclidean distance between RGB vectors was used.

Keywords: C#, image segmentation, Euclidean distance, object detection

ŽIVOTOPIS

Andrea Šušak rođena je 24. rujna 1996. godine u Slavonskom Brodu. Završila je Osnovnu školu Ivana Kozarca u Županji 2011. godine. Iste godine upisuje Prirodoslovno-matematičku gimnaziju u Županji. Nakon završetka Prirodoslovno-matematičke gimnazije i mature, 2015. godine upisuje preddiplomski studij računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku gdje trenutno pohađa treću godinu studija.

Potpis: _____

PRILOZI

Prilog 1. Na priloženom optičkom disku uz završni rad nalazi se .doc i .pdf verzija završnog rada, te izvršna datoteka i kod C# aplikacije.

Prilog 2. Kod programa

GLAVNI PROGRAM:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using AForge.Imaging.Filters;
using AForge.Imaging;

namespace AforgeNet
{
    public partial class Form1 : Form
    {

        Bitmap newFile;
        FindObject modifyRGB = new FindObject();
        FileOperations getFile = new FileOperations();
        public Form1()
        {
            InitializeComponent();
            modifyRGB.ImageFinished += OnImageFinished;

        }
        private void OpenImage_Click(object sender, EventArgs e)
        {

            newFile = getFile.OpenFile();
            DisplayImage(newFile, 3);
        }
        public void OnImageFinished(object sender, ImageEventArgs e)
        {
            DisplayImage(e.bmap, 2);
        }
        public void DisplayImage( Bitmap b, int window)
        {
            if(window==1)
            {
                pictureBox1.Image = b;
            }
            else if(window==2)
            {
                pictureBox2.Image = b;
            }
            else
```

```
            {
                pictureBox1.Image = b;
                pictureBox2.Image = b;
            }
            FindObject.Enabled = true;
        }
        private void pictureBox1_Click(object sender, EventArgs e)
        {

        }
        private void pictureBox2_Click(object sender, EventArgs e)
        {

        }
        private void FindObject_Click(object sender, EventArgs e)
        {
            modifyRGB.Segment(newFile);

        }
        private void Exit_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }
    }
}
```

KLASA FILEOPERATIONS:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Drawing;
using System.Windows.Forms;

namespace AforgeNet
{
    class FileOperations
    {
        Bitmap NewImage;

        public Bitmap OpenFile()
        {
            OpenFileDialog ofd = new OpenFileDialog();
            ofd.InitialDirectory = "D:\\Fotografije";
            ofd.Filter = "images| *.jpg; *png; *.bmp";
            if(ofd.ShowDialog()==DialogResult.OK)
            {
                NewImage = new Bitmap(Image.FromFile(ofd.FileName));
            }
            return NewImage;
        }
    }
}
```

KLASA FINDOBJECT:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Drawing;
using AForge.Imaging.Filters;
using AForge.Imaging;
using System.Drawing.Imaging;
using AForge.Math.Geometry;
using AForge;

namespace AforgeNet
{

    public class ImageEventArgs : EventArgs
    {
        public Bitmap bmap { get; set; }
    }
    class FindObject
    {
        public event EventHandler<ImageEventArgs> ImageFinished;
        protected virtual void OnImageFinished(Bitmap bmap)
        {
            ImageFinished?.Invoke(this, new ImageEventArgs() { bmap = bmap });
        }

        public void Segment(object bmp)
        {
            Bitmap bmap = (Bitmap)bmp;

            EuclideanColorFiltering filter = new EuclideanColorFiltering();
            filter.CenterColor = new RGB(Color.FromArgb(215, 10, 30));
            filter.Radius = 110;

            Bitmap objectsImage = null;
            Bitmap mImage = null;
            mImage = (Bitmap)bmap.Clone();
            objectsImage = bmap;
            filter.ApplyInPlace(objectsImage);


            BlobCounter bc = new BlobCounter();
            bc.FilterBlobs = true;
            bc.MinWidth = 5;
            bc.MinHeight = 5;
            bc.FilterBlobs = true;
            bc.ProcessImage(bmap);
            Rectangle[] rects = bc.GetObjectsRectangles();
            foreach (Rectangle recs in rects)
                if (rects.Length > 0)
                {
                    foreach (Rectangle objectRect in rects)
                    {

                        Graphics g = Graphics.FromImage(mImage);
                        using (Pen pen = new Pen(Color.FromArgb(160, 255, 160), 5))
                        {
                            g.DrawRectangle(pen, objectRect);
                        }

                        g.Dispose();
```

```
                    }

                }
            bmap = mImage;

            OnImageFinished(bmap);
        }
    }

}
```