SVEUČILIŠTE U ZAGREBU
**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

ZAVRŠNI RAD br. 3941

# Lokalizacija proizvoljno zakrenutog teksta u slikama

Filip Zelić

Zagreb, srpanj 2015.

Stranica na koju se stavlja tekst završnog zadataka

*Zahvala mentoru prof.dr.sc.Siniši Šegviću na pomoći te vrlo korisnim sugestijama tijekom izrade rada.*

# Sadržaj

# Popis slika

# 1. Uvod

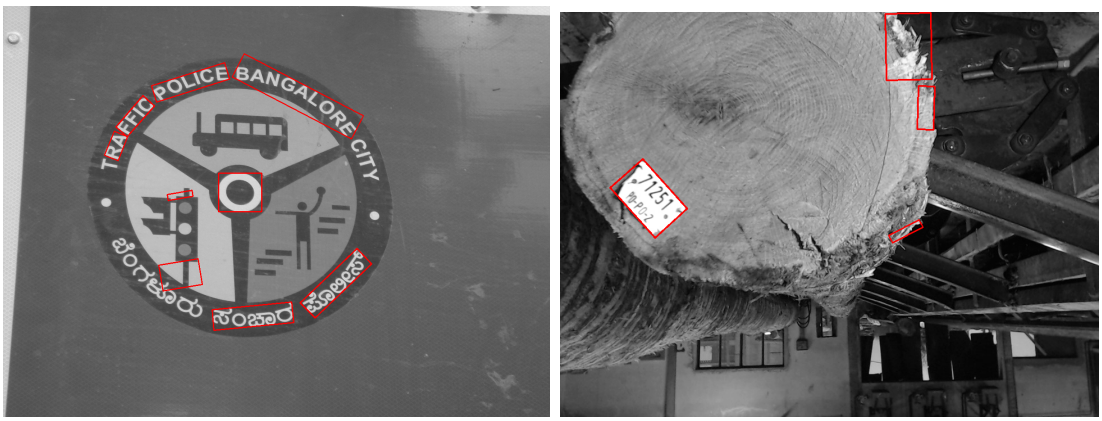
U današnje vrijeme automatsko detektiranje teksta u slikama je jedan od važnijih primjena računalnog vida. Ljudi dobivaju mnogo informacija kroz tekst te prepoznavanje može poboljšati autonomne sustave kako bi donosili bolje odluke ili jednostavno ubrzali rad na određenim zadacima. Pokazano je da efikasnost izvođenja algoritama za pretraživanje slika najviše ovisi o modulu koji se bavi njihovim otkrivanjem unutar slike. Prepoznavanje teksta se sastoji od lokalizacije teksta unutar slike nakon čega se primjenjuju metode prepoznavanja znakova. Sustavi za detekciju i prepoznavanje teksta mogu se koristiti za autonomno kretanje robota u urbanim okruženjima, aplikacijama za pomoć slabovidnim osobama, pretraživanje slika te u interakciji čovjeka i računala. Područje je vrlo interesantno i relativno mlado pa postoji puno metoda za rješavanje dobro definiranih problema. Trenutno se najviše radova fokusiralo na određivanje otprilike horizontalno pozicioniranog teksta.

U ovom radu se želi skrenuti pozornost na metode korištene u lokalizaciji proizvoljno zakrenutog teksta. Postoje tri kategorije detekcije teksta na slikama. U prvu kategoriju spadaju metode temeljene na teksturama [4]. One se koriste računanjem lokalnog intenziteta te odzivom filtra kako bi se prepoznale specijalne teksture na kojima je tekst. Najviše se koriste za detekciju horizontalnog teksta. Metode temeljene na regijama [5] prvo izvlače kandidate tekst regija kroz detekciju rubova. Nakon toga eliminiraju regije koje vrlo vjerojatno ne sadrže tekst koristeći nekoliko različitih heuristika. Treća kategorija je kombinacija metoda temeljenih na teksturama i regijama.

Cilj ovog rada je napraviti sustav za lokalizaciju proizvoljno zakrenutog teksta na slikama te usporediti nekoliko pristupa ovom problemu. Prvi pristup pri naglašavanju teksta se temelji na glađenju sive slike te oduzimanju od slike pribavljene na početku. Pod oduzimanjem se podrazumijeva oduzimanje vrijednosti intenziteta svakog piksela izvorne slike od odgovarajućeg piksela zaglađene slike. Druga metoda se koristi odzivom upravljivog filtra koji daje maksimalan odziv

filtra na sliku. Detaljnije je opisan cijeli postupak u drugom poglavlju. Nakon korištenja metode za naglašavanje teksta obavljaju se metode za dodatno postprocesiranje slike kako bi se dobile bolje detekcije teksta. Transformacija širine poteza koristi se važnim obilježjem teksta. To je činjenica da tekst ima otprilike konstantnu širinu poteza. Prema tome nad izvornom sivom slikom se prvo provodi detekcija rubova zatim se obavlja transformaciju širine poteza (Stroke Width Transform, SWT [2]). Izlaz ove transformacije je slika koja sadrži širinu poteza za rubne piksele umjesto intenziteta piksela. Dok drugi pristup za postprocesiranje je morfološka obrada binarne slike. Odabrane metode su implementirane u programskom jeziku C++ koristeći biblioteku *OpenCv*[1] koja osim što pruža sučelje za osnovne operacije dohvata i postavljanja vrijednosti piksela, također sadrži i velik broj algoritama često korištenih u području računalnog vida.

Rad je podijeljen u pet glavnih poglavlja. Nakon uvoda u 2.poglavlju dan je opis algoritama i korištenih matematičkih metoda. Ispitni skupovi slika nad kojima se provodi evaluacija predstavljeni su u 3.poglavlju. Nadalje detaljniji opis OpenCV biblioteke te organizacija programskog koda je razrađena u 4.poglavlju. Eksperimentalni rezultati su navedeni u 5.poglavlju nakon kojeg dolazi zaključak rada.




**Slika 1.1:** Primjeri detekcije teksta

[1]OpenCV (Open Source Computer Vision) je programska biblioteka namijenjena području računalnog vida i obradi slike/videa u stvarnom vremenu. Koristi se pod besplatnom open-source BSD licencom.

# 2. Algoritmi i matematičke metode

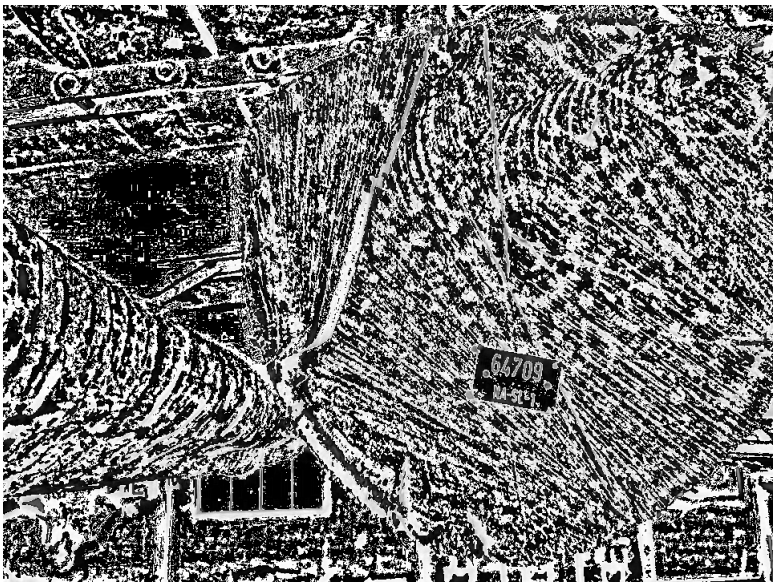
## 2.1. Metode za naglašavanje teksta

### 2.1.1. Strukturno-teksturna dekompozicija

Uklanjanje šuma sa slike je važan zadatak za obradu slika. Uglavnom se koristi kao pomoćna komponenta za veliki spektar procesa u algoritmima obrade slike. Uklanjanje šuma ćemo primjeniti na izvornim ulaznim sivim slikama kako bi dobili potpuno zamućene dijelove gdje se nalazi tekst. Koristili smo metodu *denoise_TVL1* iz programske biblioteke *OpenCV*. Na slici 2.1 prikazan je rezultat kakav bi trebali dobiti. Metoda za uklanjanje šuma ima dva parametra. Prvi je broj iteracija dok je drugi lambda. Broj iteracija je optimalno postaviti na 40. Lambda zapravo određuje jačinu efekta, njegovim smanjivanjem rezultantna slika će biti mutnija. Najbolje rezultate dobivamo za $lambdu = 0.65$.

**Slika 2.1:** Rezultat uklanjanja šuma nad izvornom slikom

Sljedeći korak je oduzeti dobivenu mutnu sliku od početne izvorne sive slike. Tako ćemo dobiti sliku na kojoj će brojke postati jače istaknute. Rezultat možemo vidjeti na slici 2.2. Metoda rezultira velikim šumom i nije korištena, ali je moguće s njom detektirati pločice na slici debla.

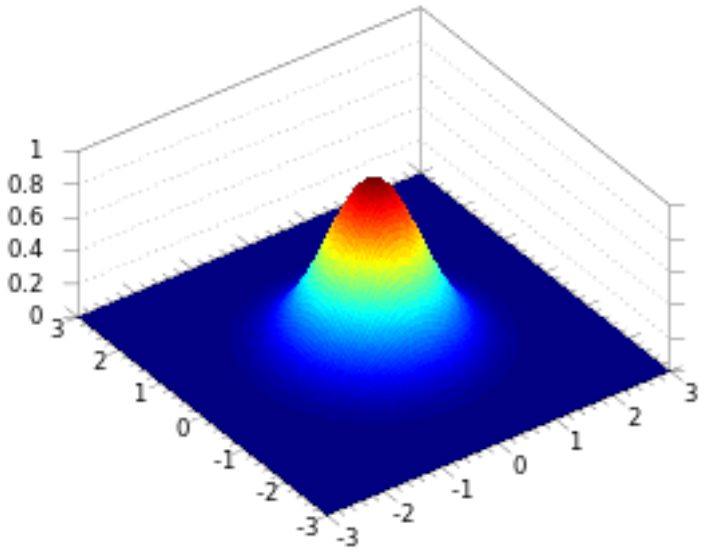


**Slika 2.2:** Rezultat nakon oduzimanja zamućene slike od izvorne sive slike

### 2.1.2. Digitalni filtar

Digitalno filtriranje je proces provođenja matematičkih operacija nad diskretiziranim signalom kako bi se potisnuli ili naglasili neki aspekti slike. Dijele se na jednodimenzionalne i dvodimenzionalne. U okviru rada koristit ćemo dvodimenzionalne filtre temeljene na drugoj derivaciji Gaussove funkcije. Dvodimenzionalna Gaussova funkcija:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.1}$$




**Slika 2.3:** Izgled dvodimenzionalne Gaussove funkcije

#### Dvodimenzionalni filtar

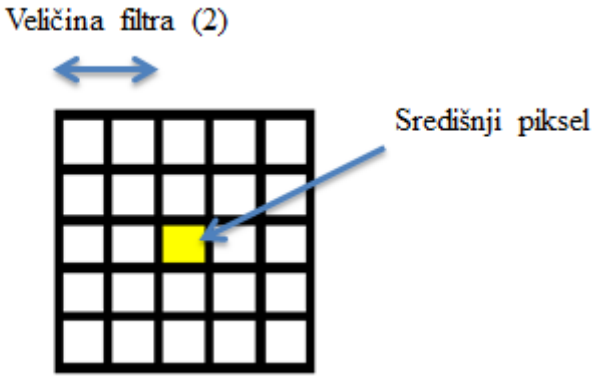
Parametar koji se predaje pri konstruiranju filtra je sigma. On utječe na veličinu filtra koji će se primijeniti na sliku prilikom računanja odziva filtra. Veličina samog filtra je udaljenost u pikselima od središnjeg piksela do krajnjeg piksela iz njegove okoline.

Povećanjem sigme povećava se promatrana okolina oko središnjeg piksela. Računa se prema formuli:

$$\text{filter\_size} = (\text{int})(2\sigma + 0.5); \tag{2.2}$$

Matrica filtra (engl. *kernel*) se određuje pomoću duljine filtra:

$$\text{matrix\_size} = 2\text{filter\_size} + 1; \tag{2.3}$$




**Slika 2.4:** Matrica filtra (engl. *kernel*)

Komponente koje su nam potrebne za računanje upravljivog filtra su druge derivacije Gaussove funkcije:

1. Filtar $G_{xx}$

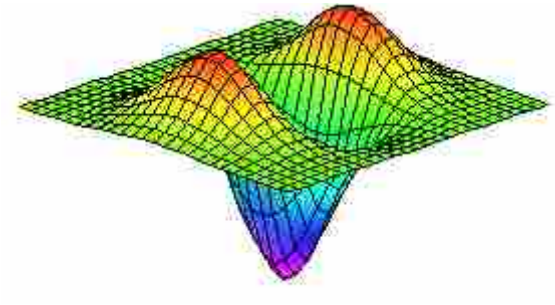
$$G_{xx}(x,y) = \frac{x^2 - \sigma^2}{2\pi\sigma^6} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.4}$$

2. Filtar $G_{yy}$

$$G_{yy}(x,y) = \frac{y^2 - \sigma^2}{2\pi\sigma^6} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.5}$$

3. Filtar $G_{xy}$

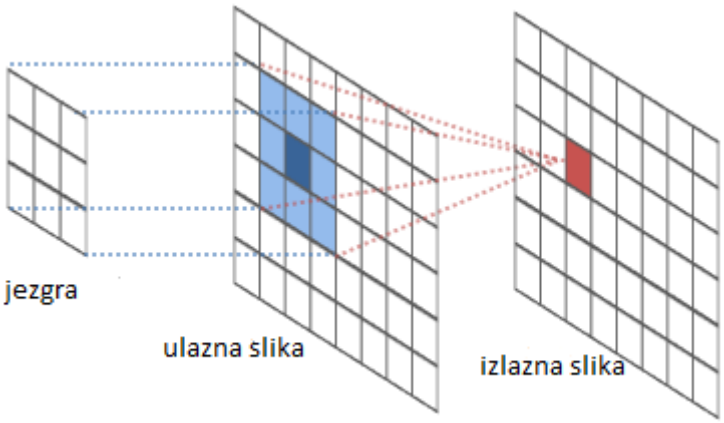
$$G_{xy}(x,y) = \frac{xy}{2\pi\sigma^6} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.6}$$



**Slika 2.5:** Usmjerena druga derivacija Gaussove funkcije $G_{xx}$

Detaljni izvod ovih komponenata nalazi se u [1]. Vrijednost ovih filtara se dobiva koristeći navedene funkcije gdje je x, udaljenost po x-osi od središnjeg piksela do piksela za koji se računa vrijednost filtra, a za y, isto jedina je razlika da se udaljenost gleda po y-osi.

Prilikom računanja odziva filtra potrebno je posebno računati vrijednost izlaznog piksela koji ovisi o skupu piksela unutar matrice filtra na ulaznoj slici. Na slici 2.6 je prikazano kako se računa odziv filtra. Vrijednost piksela se množi s odgovarajućim pikselom matrice filtra. Dobivene vrijednosti se sumiraju za cijelu matricu filtra i pohranjuju u središnji piksel na izlaznoj slici.




**Slika 2.6:** Konvolucija jezgre filtra s ulaznom slikom

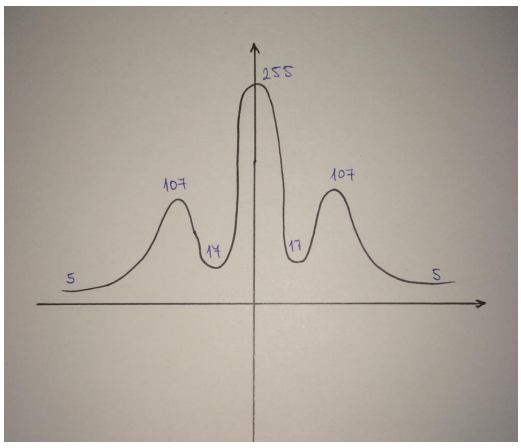
### Upravljivi filtar

Glavno svojstvo upravljivog filtra je da se odziv sastoji od intenziteta i orijentacije. Tako je moguće analitički odrediti kut[1] rotacije filtra za koji se dobiva maksimalan odziv. To će nam biti korisno za lokalizaciju proizvoljno zakrenutog teksta. Filtar $G_{xx}^{\theta}$ daje maksimalan ili minimalan odziv za neki kut rotacije. Računa se koristeći ranije spomenute komponente filtre $G_{xx}$, $G_{yy}$, $G_{xy}$ prema formuli:

$$G_{xx}^{\theta}(x,y) = G_{xx}\cos^2\theta + G_{yy}\sin^2\theta - 2G_{xy}\cos\theta\sin\theta \qquad (2.7)$$

Dobivanjem dviju vrijednosti kuta $\theta$ dobivaju se dva odziva, minimalni i maksimalni. Nakon toga se odzivi uspoređuju i u izlaznu sliku se pohranjuje maksimalan odziv. Slika 2.7 prikazuje jednostavan primjer djelovanja upravljivog filtra na sliku. Na slici 2.8 su prikazane vrijednosti piksela oko središnje bijele crte na koju djeluje upravljivi filtar.

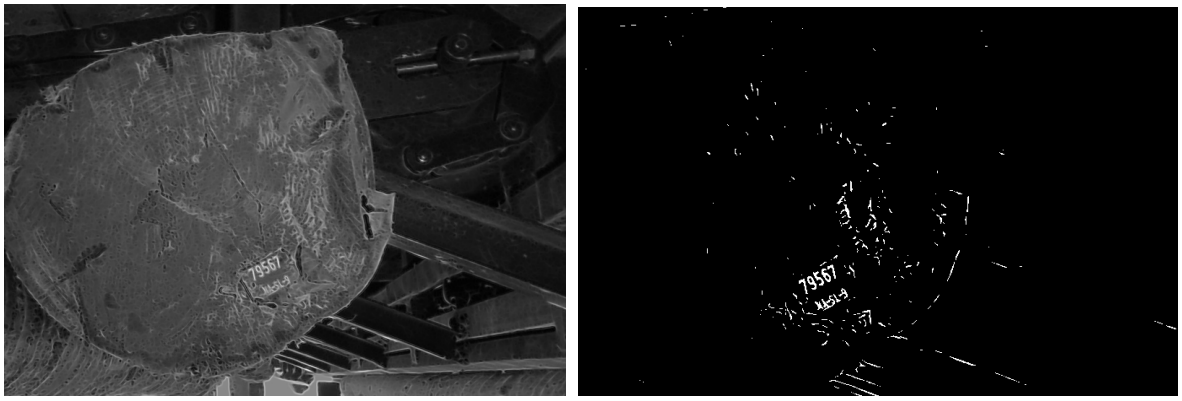
**Slika 2.7:** Djelovanje upravljivog filtra na jednostavnoj slici




**Slika 2.8:** Prikaz vrijednosti piksela oko središnje bijele crte

Slika 2.9 prikazuje djelovanje upravljivog filtra na izvornu sliku debla s pločicom. Desna slika prikazuje 1.5% najsvjetlijih piksela histograma dobivene slike jer tamo se nalaze mjesta na kojima je najveća promjena intenziteta te moguće tekstualne regije.


**Slika 2.9:** Izvorna slika debla s pločicom



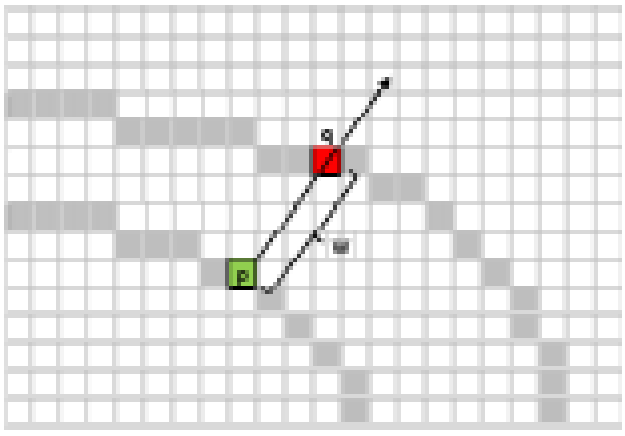

**Slika 2.10:** Lijeva slika prikazuje izlaz maksimalnog odziva, desna prikazuje rezultat nakon filtriranja 1.5% najsvjetlijih piksela iz histograma slike

Kod slika na kojima je svijetli tekst na tamnoj podlozi najveća promjena intenziteta rezultira bijelom bojom teksta na tim mjestima. S druge strane tamniji tekst na svijetloj podlozi rezultira bijelom bojom same pločice. Siva boja označava da ne dolazi do velike promjene intenziteta odnosno da je odziv jednak nuli. Upravljivi filtar pokazao se kao najbolji izbor za lokaliziranje proizvoljno zakrenutog teksta. Odziv filtra se kombinirao s prikazom samo 1.5% najsvjetlijih piksela njegovog histograma. Detekcija se mogla obaviti neovisno da li je bio tamni tekst na svijetloj podlozi ili obratno.

## 2.2. Metode za postprocesiranje slike

### 2.2.1. Transformacija širine poteza

Određivanje širine poteza (Stroke Width Transform, SWT)[2] je metoda koja pretvara slikovni podatak koji sadrži intenzitet piksela u najvjerojatniju širinu poteza. Na ovaj način moguće je detektirati tekst bez obzira na njegovo skaliranje, font, smjer i jezik. Tekst uglavnom ima fiksnu širinu poteza i zbog toga se dijelovi koji sadrže relativno konstantne širine poteza mogu grupirati u regiju (komponentu) koja vrlo vjerojatno sadrži tekst. Za provođenje algoritma potrebno je prvo učitati izvornu sliku te na njoj primijeniti detektor rubova.




**Slika 2.11:** Pikseli p i q su granični pikseli između kojih se računa udaljenost (širina poteza)[2]

### Cannyjev detektor rubova

Cannyjev detektor rubova[3] jedan je od poznatijih postupaka koji u nekoliko koraka detektira rubove u slici. Algoritam se temelji na 4 glavne faze: glađenje, izračun gradijenta, stanjivanje rubova te primjena pragova na kraju. Na početku se ulazna slika transformira u sivu sliku, nad njom se primjenjuje Gaussov filtar koji uklanja štetni šum. Uklanjanje šuma je bitno za kasniju detekciju rubova jer pomaže u eliminaciji lažnih rubova. Na izlazu ove faze dobiva se blago zamućena slika za koju se računa gradijent svakog piksela. Računa se iznos i smjer vektora gradijenta. Iznos gradijenta se dobiva kao razlika intenziteta susjednih piksela u x i y smjeru.

U trećoj fazi se provodi stanjivanje rubova do veličine jednoga piksela. Predajemo iznos te smjer gradijenta kao parametre za metodu stanjivanja. Ona na temelju njih za svaki piksel određuje treba li ga poništiti (crna boja, vrijednost piksela=0) ili ostaviti (bijela boja, vrijednost piksela=255). Završna faza Cannyjevog algoritma je usporedba s dvostrukim pragom. Pikseli s vrijednostima većim od većeg praga su valjane rubne točke, a pikseli s vrijednostima manjim od manjeg praga nisu valjanje rubne točke. Pikseli čije vrijednosti su između dva praga se dodatno promatraju. Za njih se gleda jesu li spojeni s valjanim rubnim pikselom, ako jesu tada su i oni valjane rubne točke. Na izlazu dobijemo sliku koja je binarizirana odnosno rubni pikseli su prikazani bijelom bojom dok je ostatak piksela crne boje. Dobivenu sliku Cannyjevim detektorom rubova *edge_Image* prosljeđujemo metodi za izračun širine poteza.

**Slika 2.12:** Rezultat Cannyjevog detektora rubova nad ulaznom slikom

### Određivanje širine poteza

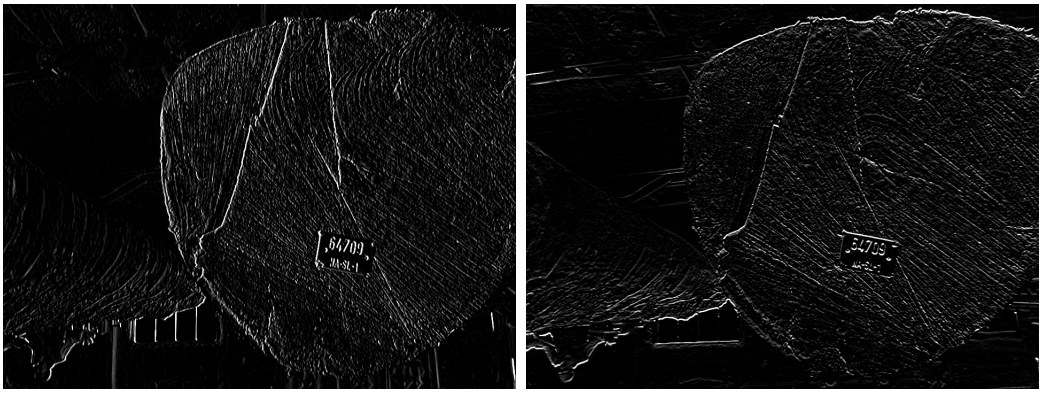
Računanje širine poteza zahtijeva izračunavanje gradijenta slike u x i y smjeru jer će biti potreban smjer gradijenta svakog piksela. Gradijent slike je vektorsko polje u smjeru najvećeg mogućeg rasta intenziteta piksela. Duljina vektora gradijenta odgovara količini promjene u tom smjeru. Velika promjena intenziteta rezultirat će velikom duljinom vektora gradijenta. Gradijent slike:

$$\nabla f = \frac{\partial f}{\partial x}\hat{x} + \frac{\partial f}{\partial y}\hat{y}$$

Smjer gradijenta se računa prema sljedećoj formuli:

$$\theta = \arctan \frac{\frac{\partial f}{\partial y}}{\frac{\partial f}{\partial x}}$$

Na slici 2.13 prikazane su slike gradijenta u x te y smjeru dobivene iz izvorne sive slike. Metoda transformacije širine poteza kao parametre prima sliku obrađenu Cannyjevim detektorom te slike gradijenta u x i y smjeru.

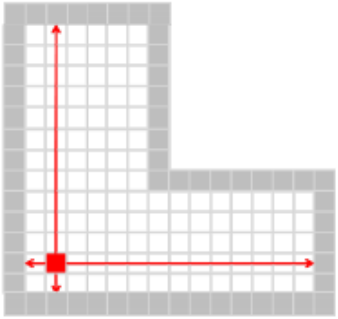


**Slika 2.13:** Slike gradijenta u x i y smjeru

**Algoritam transformacije možemo opisati u koracima:**

1. Vrijednost svakog piksela izlazne slike transformacije postavljamo na $\infty$ (programski smo postavili na -1)
2. Promatra se smjer gradijenta $d_p$ svakog piksela p koji se nalazi na rubu
3. Ako p leži na granici poteza tada $d_p$ mora biti otprilike okomit na orijentaciju poteza (smjer promjene intenziteta)
4. Pratimo zraku $r = p + nd_p$ , $n > 0$ dok ne dospijemo do drugog rubnog piksela q
5. Nakon što dospijemo do piksela q tada promatramo $d_q$, smjer gradijenta piksela q. Ako je $d_q$ otprilike suprotan u odnosu na $d_p$ tj. provjerava se vrijedi li jednakost $d_q = -d_p \pm \frac{\pi}{6}$
6. Ako jednakost vrijedi tada svaki element na segmentu $[p, q]$ dobiva vrijednost udaljenosti $w = \|\mathbf{p} - \mathbf{q}\|$ ako već nema manju vrijednost
7. U slučaju da ne vrijedi jednakost ili se ne pronađe q rubni piksel od početnog p piksela zraka r se odbacuje

Zbog nekih posebnih situacija kao na slici 2.14 neće se dobro izračunati širina poteza. Takve situacija mogu biti česte kod tekstova i brojki. Tada prolazimo po neodbačenim zrakama r ponovno te računamo medijan pohranjenih vrijednosti tih piksela zrake i postavljamo ih na vrijednost jednaku njihovom medijanu ili vrijednost većoj od medijana. Metoda nije davala dobre rezultate zbog čega smo koristili morfološku obradu binarne slike

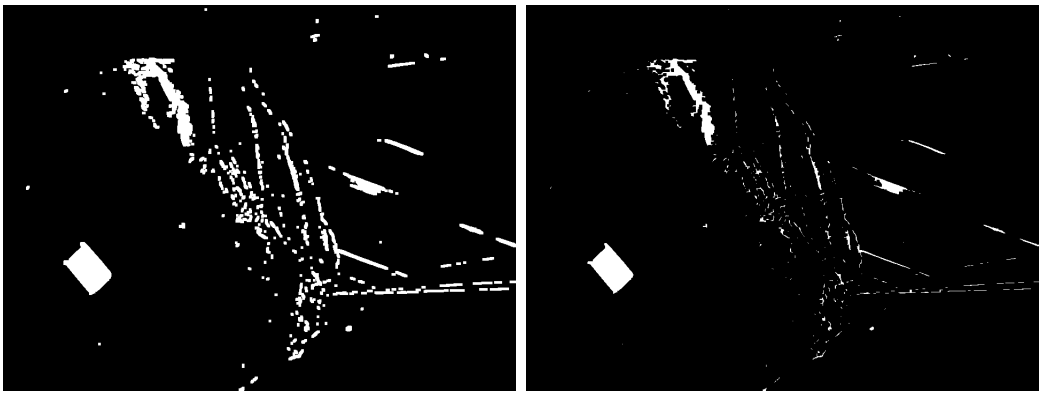


**Slika 2.14:** Slučaj kada širina poteza neće biti dobro izračunata

### 2.2.2. Morfološka obrada binarne slike

Za poboljšanje odziva vršimo postupak "zatvaranja" koji se sastoji od dva postupka, dilatacije i erozije. Prvo se vrši dilatacija, zatim erozija zadane binarne slike. Dilatacije je postupno proširenje regija piksela. Erozija je erodiranje piksela koji se nalaze na granicama. Na slici 2.16 lijevo je prikazan postupak dilatacije, a desno je postupak erozije. U programskoj implementaciji korištene su metode iz OpenCV biblioteke dilate i erode.

**Slika 2.15:** Binarna slika na kojoj se primjenjuje morfološka obrada



**Slika 2.16:** Prikaz postupaka dilatacije i erozije

# 3. Ispitni skup slika

Skup ispitnih snimki za evaluaciju navedenih metoda sadrži nekoliko datoteka po 80 slika u svakoj datoteci. Slike prikazuju debla na kojima se nalazi pločica sa serijskim brojem. Pločice se nalaze proizvoljno zakrenute na deblu i poslužit će kao dobar skup slika za ispitivanje efektivnosti razvijene metode. Boja pločice i teksta se mijena, što za neke slučajeve otežava evaluaciju. Zbog toga možemo podijeliti slike na one u kojima je serijski broj tamne boje na svijetloj podlozi pločice i na one gdje je serijski broj bijele boje, a podloga tamne boje. Rezolucija slika je 960x720 piksela.

**Slika 3.1:** Primjer slike svijetlog teksta na tamnoj pločici

**Slika 3.2:** Primjer slike tamnog teksta na svijetloj pločici

Drugi skup ispitnih snimki prikazuje snimke teksta u urbanoj okolini. Tekst na slikama se nalazi proizvoljno zakrenut i pojavljuje se nekoliko tekstualnih regija. Rezolucija slika varira od 1280x644 do 1280x960 piksela.


**Slika 3.3:** Primjeri drugog ispitnog skupa slika

# 4. Programska izvedba i vanjske biblioteke

U sljedećim poglavljima opisana je programska implementacija postupaka. U radu je korišten programski jezik C++ te biblioteka OpenCV. Odabran je jezik C++ zbog ostvarivanja veće brzine te kompatibilnosti s postojećim modulima.

## 4.1. Biblioteka OpenCV

Biblioteka OpenCV(engl. *Open Source Computer Vision Library*) je biblioteka otvorenoga koda koja sadrži preko 2500 optimiziranih algoritama iz područja računalnog vida. Podržana je na svim značajnijim operativnim sustavima: Windows, Linux i Mac OS te postoje inačice biblioteke namijenjene za korištenje u nekoliko programskih jezika: C, C++, Python, Java te Matlab. OpenCV je distribuiran pod licencom BSD[1] što omogućuje slobodnu uporabu u akademskim programima.

Ima modularnu strukturu što znaci da uključuje više dijeljenih i statičkih biblioteka. Primjer glavnih modula:

* **core** - funkcionalnost jezgre, osnovne strukture i algortimi
* **imgproc** - algoritmi procesiranja slika
* **highgui** - korisničko sučelje za prikaz rezultata i slika
* **video** - obrada videa te estimacija gibanja

[1]BSD je obitelj slobodnih softverskih licenci, uz nametanje minimalnih ograničenja preraspodjele

## 4.2. Implementacija upravljivog filtra

Razred *SteereableFilter* ima podatkovni član sigma koji određuje širinu promatrane okoline oko središnjeg piksela odnosno veličinu matrice filtra koju ćemo primijeniti nad slikom. U nastavku je dan popis korištenih metoda iz modula te opisi parametara.

```
void steerable_filter(Mat &image,Mat &result_image, double sigma)
```

Unutar OpenCV biblioteke za pohranu slike koristi se struktura Mat. Metoda *steerable_filter* prima izvornu sliku, sliku u koju će pohraniti rezultat te parametar sigma. U prvom dijelu računaju se filtri $G_{xx}, G_{yy}$ i $G_{xy}$ nakon čega se filtri primjenjuju na izvornu sliku. Na temelju izračuna odziva filtara se određuje minimalni i maksimalni kut theta. Potom se dobiveni kutevi theta predaju metodi *calculateGTheta* koja računa odziv prema dobivenim kutevima.

```
double calculateGTheta(double Gxx,double Gyy,
double Gxy, double theta)
```

Parametri su *double* vrijednosti filtara i izračunati kut theta. Nakon poziva ove metoda za dva kuta određuje se koji je odziv minimalni, a koji je maksimalni. Maksimalni odziv filtra sprema se u *result_image* sliku koja je ujedno i izlaz ovog modula.

## 4.3. Implementacija transformacije širine poteza

Razred *strokeWidthTransform* prima izvornu sivu sliku te obavlja detekciju rubova Cannyjevim detektorom.

Izlazna slika detektora rubova pohranjuje se u *edgeImage*. Pronalazi se gradijent u x i y smjeru izvorne slike. Rezultantna slika *swt_image* prije obrade postavlja sve vrijednosti na $-1$. Navedene parametre predajemo glavnoj metodi koja obavlja transformaciju.

```
void strokeWidthTrasnformation(Mat &edgeImage,
Mat &swt_image,std::vector<Ray> &rays, Mat &grad_x, Mat &grad_y)
```

Metoda pomoću dobivenih gradijenata i slike rubova pronalazi širine poteza te ih pohranjuje u kolekciju zraka. Svaka zraka se sastoji od početnog rubnog piksela p, završnog rubnog piksela q te piksela između njih u smjeru gradijenta.

U piksele zrake pohranjuje se vrijednost širine umjesto intenziteta. Nakon toga se poziva metoda *MedianFilter* koja prolazi po neodbačenim zraka i računa medijan pohranjenih vrijednosti. Postavlja sve piksele unutar zrake na vrijednost jednaku medijanu ili vrijednost veću od medijana.

```
void MedianFilter(Mat &swt_image,
                  std::vector<Ray> &rays)
```

Izlazna slika je rezultat transformacije i pohranjena je u *swt_image*.

## 4.4. Implementacija strukturno-teksturne dekompozicije

Razred *ImageSubtraction* sadrži dva podatkovna člana. Prvi je broj iteracija *niters*, a drugi *lambda*. Oba parametara određuju jačinu zaglađenja izvorne sive slike. Najbolje rezultate dobivamo za $niters = 40$ i $lambda = 0.65$.

```
denoise_TVL1(observations, denoised_image, lambda, niters)
```

Metoda *denoise_TLV1* sprema zaglađenu sliku u *denoise_image*. Slika je zamućena toliko da se brojke potpuno izgube.

```
void subtract_images(Mat& image, Mat& denoised_image, Mat& minus_image)
```

Sljedeći korak je poziv metode koje obavlja oduzimanje zamućene slike od izvorne slike. Rezultat sprema u *minus_image*.

## 4.5. Implementacija binarizacije prema histogramu slike

Program namijenjen binariziranju slike prema izračunatom histogramu će nam poslužiti kako bi nakon obrade slike dodatno istaknuli dijelove slike gdje se nalazi tekst. Ideja je da se prikaže određeni postotak najsvjetlijih piksela, ali tako da određeni prag na sve slike podjednako djeluje. Prvo prikazujemo izračun histograma slike.

```
// popunjavanje histograma
for(int y=0; y < image.rows; ++y) {
        for(int x=0; x < image.cols; ++x) {
                histogram[ (int)saveImage.at<uchar>(y,x) ]++ ;
        }
}
```

Slijedi poziv metode određivanja granice iz histograma kojoj kao parametre šaljemo ukupni broj piksela slike, dobiveni histogram i postotak prikaza najsvjetlijih piksela.

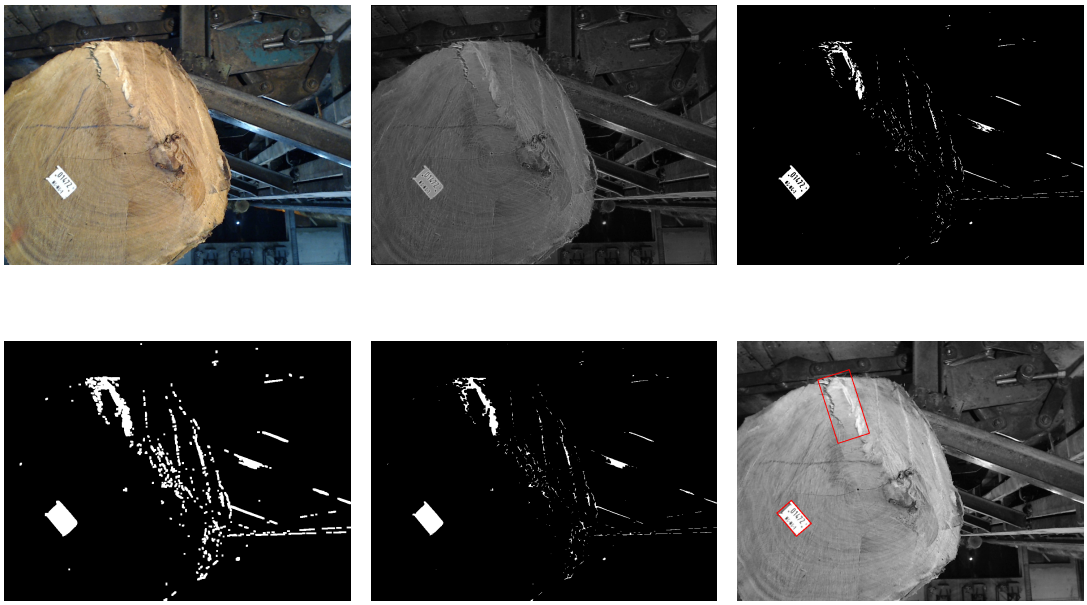
```
        // odredivanje granice prema postotku histograma
int findThreshold(int histogram[], int total, int percentage) {
        int curr_sum = 0;
        int threshold = 0;
        int total_pixels = total*percentage;
        for(int i=0; i<256; ++i) {
                if(curr_sum < total_pixels) {
                        curr_sum += histogram[i];
                } else {
                        threshold = i ;
                        return threshold;
                }
        }
}
```

Nakon izračuna granice iz histograma prelazimo po slici te intenzitet svakog piksela uspoređujemo s izračunatom granicom. Piksele s vrijednostima manjim od granice postavljamo na vrijednost 255. Rezultat metode prikazuje određeni postotak najsvjetlijih piksela bijelom bojom na crnoj podlozi.

# 5. Eksperimentalni rezultati

## 5.1. Cijeli proces obrade slike

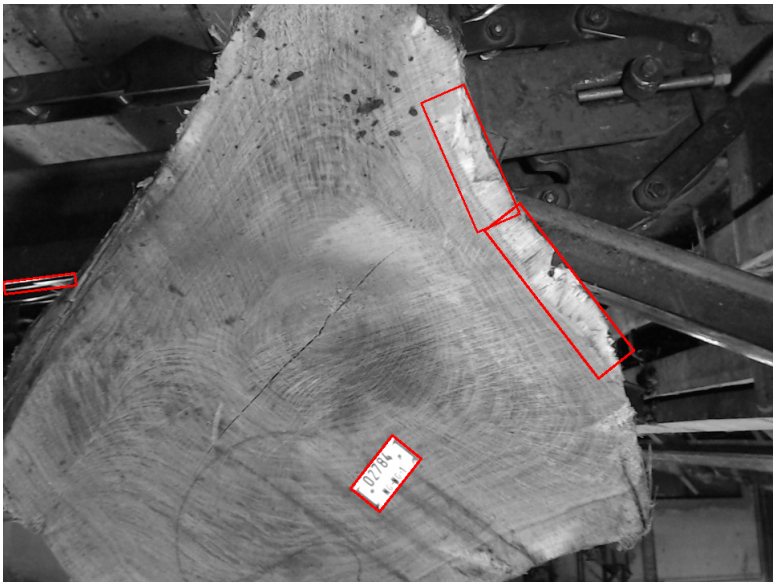
Glavni cilj rada bio je pronaći proizvoljno zakrenuti tekst ili brojke na slici. Na slikama je prikazan cjelokupni proces obrade slike po koracima. Učitavamo izvornu sliku, pretvaramo je u sivu slike te nad njom primjenjujemo upravljivi filtar. Druga slika prikazuje rezultat upravljivog filtra. Nakon toga prikazujemo samo 1.5% najsvjetlijih piksela histograma slike filtra. Proces zatvaranja se primjenjuje na dobivenu slike i rezultat se vidi na četvrtoj i petoj slici. Za dobivanje detekcije na sliku obrađenu dilatacijom i erozijom primjenjujemo pronalaženje minimalnih zarotiranih pravokutnika pomoću metode minAreaRect. Rezultat detekcija je prikazan na zadnjoj slici.



**Slika 5.1:** Proces obrade slike po koracima

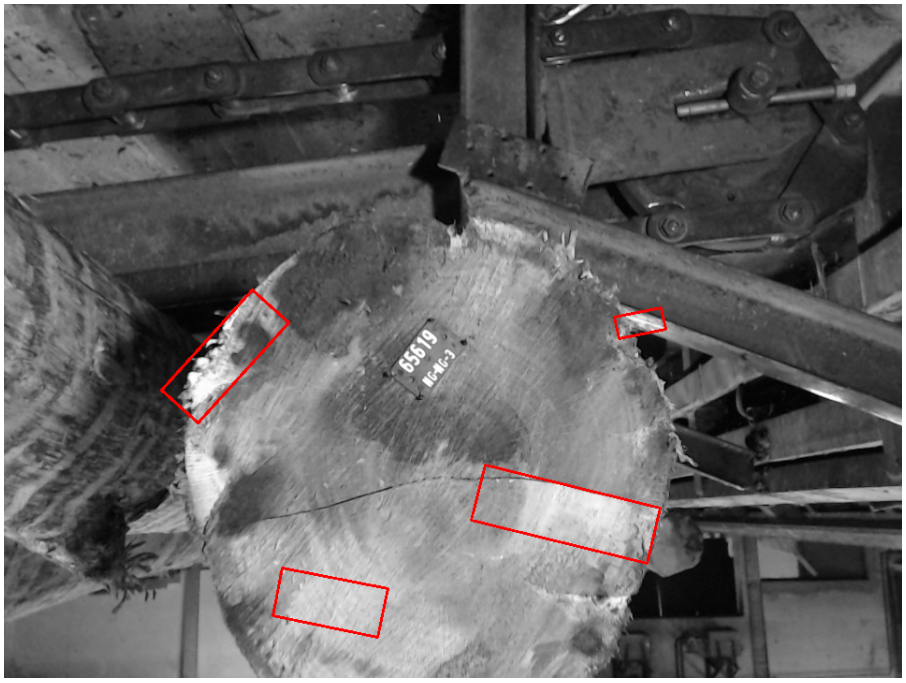
## 5.2. Ispitni skup snimki debla s pločicom

Pločice koje numeriraju debla mogli bi podijeliti na pločice koje sadrže tamni tekst na svijetloj podlozi te one koje sadrže svijetli tekst na tamnoj podlozi. Metode strukturno-teksturna dekompozicija te transformacija širine poteza nisu davale zadovoljavajuće rezultate. Većina slika dobivene tim metodama predstavljale su šum. Koristeći metodu upravljivog filtra dobili smo najbolje rezultate te pomoću nje smo evaluirali dva skupa slika. Nije bilo potrebno odvajati pločice u dva zasebna skupa. Na slici 5.2 vidimo prikaz detekcije pločice s brojkama kada je tamni tekst na svijetloj pločici te obratno.



**Slika 5.2:** Primjer lokalizacije slike tamnog teksta na svijetloj pločici

Na slikama se pojavljuju krive detekcije (engl. *false positive*), ali to neće biti veliki problem jer pri integraciji sa sustavom za prepoznavanje teksta jednostavno prođemo po svim detekcijama i vrlo lako pronađemo na kojoj je tekst. Veći problem predstavlja propuštena detekcija odnosno slučaj kada se uopće ne pronađe pločica s brojkama (engl. *false negative*). Propuštena detekcija se javlja rijetko te u specifičnim slučajevima kad osvjetljenje smanji kontrast brojki i pločice. Jedan takav primjer vidimo na sljedećoj slici 5.3.




**Slika 5.3:** Primjer kada nema niti jedne pozitivne detekcije pločice s brojkama

Broj uspješnih detekcija najbolji je za parametar $\sigma = 1.2$. Veličina filtra matrice tada je $5x5$ i filtar tada ima najjači odziv na debljinu brojki koje se nalaze na pločici. Rezultate smo procijenili srednjom vrijednosti odziva, preciznosti te vremena izvođenja. Odziv i preciznost se računaju prema sljedećim formulama:

$$Preciznost = \frac{Točne\ detekcije}{Točne\ detekcije + Lažne\ detekcije} \tag{5.1}$$

$$Odziv = \frac{Točne\ detekcije}{Točne\ detekcije + Propuštene\ detekcije} \tag{5.2}$$

Dobiveni rezultati nakon obrade 40 slika debla iz prvog skupa slika:

* Preciznost: 29.3%
* Odziv: 82.5%
* Vrijeme izvođenja: 0.545s

Preciznost je nešto niža jer postoji samo jedna pločica na slici i tada je brojnik jedan u većini slučajeva, a nazivnik varira ovisno o slici. Veliki broj slika ima

prozor unutar slike koji je gotovo uvijek kriva detekcija. Osvjetljenje prozora potpuno se razlikuje od intenziteta ostatka prostorije što dovodi do lažne detekcije. No nama je bitniji parametar odziv jer je bitno da na svakoj slici nema propuštenih detekcija. Rezultat od 82.5% za odziv je vrlo dobar, ali naravno mogao bi se poboljšati daljnjim razvojem metode. Vrijeme izvođenja pokazuje da bilo moguće napraviti sustav koji u realnom vremenu obavlja lokalizaciju i detekciju kada bi se dodatno optimirao program.

## 5.3. Ispitni skup snimki u urbanoj okolini

Drugi skup slika prikazuje nekoliko tekstualnih regija unutar slika u ubranoj okolini gdje je puno kompleksnija pozadina. Odabrali smo 20 slika s jednostavnijim pozadinama za provođenje evaluacije. Razlog je kompleksna pozadina koja jako utječe na mogućnost pronalaska teksta. Na primjer lišće uzrokuje puno lažnih detekcija što vidimo iz slike 5.4. Velika promjena intenziteta pri izmjeni lišća i svjetlosti uzrokuje te detekcije.


**Slika 5.4:** Primjer utjecaja kompleksne pozadine na pronalazak teksta u sceni

Dobiveni rezultati nakon obrade 20 slika iz drugog skupa slika:

* Preciznost: 60.6%
* Odziv: 75.1%
* Vrijeme izvođenja: 0.918s

Rezultati za preciznost su u ovom slučaju bolji dok odziv je niži jer postoji puno više tekstualnih komponenti koje je program detektirao i više propuštenih detekcija. Vrijeme izvođenja je veće u odnosu na rezultate slika debla jer postoji više tekstualnih regija koje je potrebno iscrtati. Na slici 5.5 i 5.6 vidimo primjere iz tog skupa na kojima je provedena uspješna detekcija.


**Slika 5.5:** Primjer uspješne detekcije nad drugim skupom slika


**Slika 5.6:** Primjer uspješne detekcije nad drugim skupom slika

# 6. Zaključak

U ovom radu predstavili smo metode za lokalizaciju proizvoljno zakrenutog teksta u slici. Pronalazak takvih metoda uvelike bi poboljšale mogućnost razumijevanja okoline iz slike. Zadatak na koji smo se fokusirali je bilo očitavanje serijskih brojeva s pločice koja se nalazi na deblima te pronalaženje teksta u slici. Obradili smo nekoliko načina na koji bismo mogli doskočiti problemu zakrenute lokalizacije teksta. Metodu koja koristi upravljivi filtar smo koristili pri evaluaciji ispitnih slika u kombinaciji s binarizacijom prema histogramu slike. Temelji se na korištenju upravljivog filtra koji daje maksimalni odziv neovisno o položaju. Odziv je najvažniji parametar rezultata jer pri integraciji sustava lokalizacije i detekcije bitno je minimizirati propuštene detekcije. Srednja vrijednost odziva dobivena nad slikama debla iznosi 82.5% dok nad slikama teksta iznosi 75.1%. Metoda strukturno-teksturne dekompozicije te transformacija širine poteza nisu davali vrlo korisne mogućnosti daljnjeg razvoja.

U budućem bi se radu uspješnost detekcije mogla poboljšati kombinacijom nekih dodatnih metoda segmentacije slike koje bi s većom sigurnošću omogućile lokaliziranje teksta. Sljedeći koraci nakon poboljšanja metode lokaliziranja bi bila integracija sa sustavom za prepoznavanje znakova. Takav povezani sustav mogao bi automatizirati procese očitavanja stanja skladišta.

# Literatura


[1] A.Majić. *Pronalaženje prometnog traka korištenjem upravljivih filtara.* Završni rad, Sveučilište u Zagrebu, Fakultet Elektrotehnike i računarstva, 2009.

[2] Y.Wexler B.Epshtein, E.Ofek. *Detecting Text in Natural Scenes with Stroke Width Transform.* Microsoft Corporation, 2010. URL `http://www.math.tau.ac.il/~turkel/imagepapers/text_detection.pdf`.

[3] Y.Xia Q.Wang P.Zhou, W.Ye. *An Improved Canny Algorithm for Edge Detection*, 2011. URL `http://www.jofcis.com/publishedpapers/2011_7_5_1516_1523.pdf`.

[4] A.Yuille X.Chen. *Detecting and Reading Text in Natural Scenes.* 2004.

[5] S.Goto & T.Ikenaga Y.Liu. *A Contour-Based Robust Algorithm for Text Detection in Color Images*, 2006. URL `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.96.5641&rep=rep1&type=pdf`.

**Lokalizacija proizvoljno zakrenutog teksta u slikama**

**Sažetak**

U sklopu rada razmatra se lokalizacija proizvoljno zakrenutog teksta u slikama. Opisana su dva pristupa naglašavanja teksta među kojima se najuspješnijim pokazao upravljivi filtar. On pronalazi maksimalni odziv neovisno o položaju. Nad slikama nakon uporabe upravljivog filtra obavljena je binarizacija prema pragu dobivenom iz histograma. Morfološka obrada binarne slike se koristi kako bi se poboljšalo pronalaženje teksta. Metode su evaluirane nad skupom slika debla označenih pločicom sa serijskim brojem i skupom slika tekstualnih komponenti u ubranoj okolini. Programski kod pisan je u jeziku C++ uz korištenje biblioteke OpenCV. Rezultantne slike prikazane su u poglavlju o eksperimentalnim rezultatima.

**Ključne riječi:** lokalizacija teksta, transformacija širine poteza, upravljivi filtar, proizvoljno zakrenuti tekst, računalni vid, Cannyjev detektor rubova, OpenCV, C++

**Localization of arbitrarily rotated text in images**


## Abstract

This paper considers localization of arbitrarily rotated text in images. Several approaches are described and most successful one is use of steerable filter. Steerable filter finds the maximum response regardless of the position. Morphological processing of binary image is used after applying steerable filter in the detection of the rotated text. Code is written in C++ programming language with use of OpenCV library. Methods were evaluated against a set of images of tree trunks marked with the serial number plate and set of images with textual component in urban environment. Results and result images are shown in the section on experimental results.



**Keywords:** text localization, stroke width transform, steerable filter, arbitrarily rotated text, computer vision, Canny edge detector, OpenCV, C++