Computer vision-based detection of roadside vegetation using features from the visible spectrum

Harbaš, Iva

Doctoral thesis / Disertacija

2019

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Electrical Engineering and Computing / Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:168:215053

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-09-05**

Repository / Repozitorij:

FER Repository - University of Zagreb Faculty of Electrical Engineering and Computing repozitory

DIGITALNI AKADEMSKI ARHIVI I REPOZITORIJI

University of Zagreb

FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

Iva Harbaš

Computer vision-based detection of roadside vegetation using features from the visible spectrum

DOCTORAL THESIS

Zagreb, 2019.

University of Zagreb

FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

Iva Harbaš

Computer vision-based detection of roadside vegetation using features from the visible spectrum

DOCTORAL THESIS

Supervisor: Associate Professor Marko Subašić, PhD

Zagreb, 2019.

Sveučilište u Zagrebu

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Iva Harbaš

Računalna detekcija vegetacije uz prometnice temeljena na značajkama iz vidljivog dijela spektra

DOKTORSKI RAD

Mentor: Izv. prof. dr. sc. Marko Subašić

Zagreb, 2019.

This doctoral thesis was made at University of Zagreb Faculty of Electrical Engineering and Computing, at the Department of Electronic Systems and Information Processing.

Supervisor: Associate Professor Marko Subašić, PhD

Doctoral thesis contains: 86 pages

Doctoral thesis num.: ___________

About the Supervisor

Marko Subašić was born in 1976. He graduated, got his MSc degree and Ph.D. degree from the Faculty of Electrical Engineering and Computing at the University of Zagreb in 1999, 2003 and 2007 respectively. Since 1999 he has been working at the Department for Electronic Systems and Information Processing at the Faculty of Electrical Engineering and Computing at the University of Zagreb, and is currently working as an Associate Professor. at FER, he teaches several courses at graduate and undergraduate studies. He was a mentor for more than 30 graduate and undergraduate theses. His research interests are in image processing and analysis, and neural networks with a particular interest in image segmentation and detection techniques, and deep learning. He is a member of the Image Processing Group at the Department of Electronic Systems and Information Processing. Dr. Subašić is a member of IEEE - Computer Society, Croatian Center for Computer Vision, Croatian Society for Biomedical Engineering and Medical Physics, Centre of Research Excellence for Data Science and Advanced Cooperative Systems. Dr. Subašić actively participated in the organization of several international conferences. He participates and has participated in several research projects funded by national and EU agencies.

Thanks and Dedication

Even though my name is on the front page I am not the only one creditable for finishing this PhD journey I started almost 7 years ago.

I might have done the grunt work of coding, testing and writing the thesis, but there have been several individuals that helped me along the way. Some help came in the form of technical knowledge and discussion while some consisted of motivation and encouragement which was sometimes even more important.

To start off, I would like to thank my mentor Marko who is the one that made it possible for me to work on the Faculty for two years and do most of the research necessary for my thesis. Also, thank you for spending all those hours debugging code with me and for all the motivational speeches and emails you sent. If I did not show it at the time, they did help and I am grateful.

Next, I would like to thank my colleague and friend Pavle who shared his woking space with me for two years and who provided advice, valuable insight that was irreplaceable and all kinds of stories to pass the time.

Since I left the Faculty I basically stopped working on my thesis and for a long time, it was standing still. This changed when I started working at Ericsson Nikola Tesla where Goran Molnar made a special effort to jump-start me into a ’PhD mode’. I wouldn’t be writing this ’Thanks and dedication’ section now if it wasn’t for him, and all the support I got from Emina, Darko, Zoran, Mario, Vlasta and Mardji who provided me with time and resources when I needed them to work on my thesis.

A special thanks here must go to my team at work - JARvis. A person could not have wished for a better team and I do not say this because they were super OK with me taking the time and working on my thesis, but for the everyday shenanigans that make coming to work a fun, exciting and challenging experience. So, a very special thanks to Dino, Dario, Zrinka, Ivan, and Matej.

Thank you Eva for all the talks we had where we tried and motivate and comfort each other - now we can finally get together and not talk about our PhDs.

Thank you Vedrana for having your door always open - but now, our watch has ended.

Thank you Joke for asking about my PhD often enough that I just had to finish it - now I have a different answer ready for you.

Thank you Mirkać for listening to me even if you did not understand - I hope that this will motivate you to do your best.

One very special ’Thank you’ is reserved for a very special person, who is with me every day, who knows everything that troubles me and everything that makes me happy. There were a lot of ups and downs in the making of this thesis and he is the one who always knew what to say to make me come back to this over and over again and who never gave up on me when I would be in a rut and couldn't see the end. There are no words how I can thank you, my partner on the mats, off the mats, in life, and in crime - thank you Nikola, for being there no matter what.

And finally, I dedicate this to my parents who were more nervous and excited about this then I was. The look on your faces when the defense was over was priceless and I am very happy that I was able to make you so happy and proud. Thank you Dad for constantly asking me "How is the PhD coming along?", and thank you Mom for keeping the balance and not mentioning it as often.

Thank you both for being awesome parents and for all the support. Without you two I wouldn't be where I am now, in every sense possible.

Abstract

This thesis contains a comprehensive analysis and discussion of issues and solutions encountered while developing a method for detecting roadside vegetation. The goal was to develop a method which would complement vehicles used for roadside maintenance. This would provide an opportunity to automate such tasks which are currently manually operated by maintenance workers. Vegetation detection is a common theme in the area of remote sensing where vegetation is detected from satellite images and used to monitor the health of vegetation which is affected by urbanization and industrialization. Most of vegetation detection methods are based on using satellite images which contain information from the spectrum that is invisible to the human eye. Our work focuses on developing a method based on using features from the visible part of the spectrum for detecting various vegetation types using a simple camera mounted on the vehicle. An extensive overview of developed methods is given starting with simpler approaches based on different manually selected features for machine learning and ending with a method based on deep learning where features are learned. We have shown that Fully Convolutional Neural Networks can be effectively used in a real world application for detecting roadside vegetation. For training and testing purposes, we have created our own image database which contains roadside vegetation in various conditions. Promising experimental results are presented with a discussion of encountered problems in real-world application as well as a comparison with several alternative approaches.

Keywords: Image processing, vegetation detection, roadside maintenance, machine learning, convolutional neural networks

Prošireni sažetak

Računalna detekcija vegetacije uz prometnice temeljena na značajkama iz vidljivog dijela spektra

U ovom radu je prikazana sveobuhvatna analiza metoda za detekciju vegetacije uz prometnice te rasprava o problemima i rješenjima koja su se susrela prilikom razvoja. Cilj je bio razviti metodu koja će poslužiti kao pomoć vozilima koja se koriste za održavanje cesta. Opremanje strojeva računalnim vidom i sposobnostima strojnog učenja omogućilo bi automatizaciju takvih zadataka koji se trenutno obavljaju ručno.

Detekcija vegetacije uobičajena je tema u području daljinskih istraživanja gdje se vegetacija detektira u satelitskim snimkama.
Detekcija i analiza vegetacije iz satelitskih snimaka se koristi za praćenje zdravlja vegetacije na koju utječu ubrzana urbanizacija i industrijalizacija. Osim različitih primjena u daljinskom istraživanju, nedavno je došlo do razvoja metoda za detekciju vegetacije u robotici gdje se takve metode koriste kao dodatak navigaciji autonomnih vozila koja su namjenjena da operiraju u ne-urbanim sredinama i u sredinama zaraslima vegetacijom.
Vegetacija se smatra preprekom koju autonomno vozilo može preći, kao npr.visoka trava, za razliku od drugih, krućih prepreka koje bi vozilo moralo zaobići.

Većina metoda za detekciju vegetacije temelji se na korištenju satelitskih snimaka koje sadrže podatke iz spektra koji je nevidljiv ljudskom oku.

Uzimajući to u obzir, metode za detekciju vegetacije se mogu podijeliti u dva područja ovisno o tipu korištenih značajki i to na:

- Metode bazirane na značajkama iz vidljivog dijela spektra
- Metode bazirane na značajkama iz nevidljivog dijela spektra

Većina metoda zasnivanih na korištenju satelitskih snimaka koristi značajke iz nevidljivog dijela spektra budući da sateliti snimaju Zemljinu površinu u raznim modalitetima, a i pokazalo se da je infracrveni dio spektra iznimno korisan za detekciju vegetacije jer vegetacija bogata klorofilom reflektira najviše elektromagnetskog zračenja baš u tom području spektra.

S druge strane, metode razvijene na ne-satelitskim snimkama se često baziraju na kombiniranju značajki iz vidljivog i nevidljivog dijela spektra. Također, neki istraživači koriste i posebnu opremu poput LADARa koji snimaju 3D okruženje iz kojeg se računaju 3D značajke (npr. teksture).

Ideja iza ovog rada je bila razvijanje metode koja bi se bazirala na korištenju značajki isključivo iz vidljivog dijela spektra što bi se postiglo montiranjem obične kamere na vozilo te nebi zahtjevalo nikakvu modifikaciju kamere niti posebnu dodatnu opremu.

Rad je organiziran u četiri velika poglavlja i to:

- **’Uvod’**

Uvod detaljnije opisuje motivaciju iza teme i ideje za potencijalno korištenje metode za detekciju vegetacije u stvarnim primjenama te daje kratki pregled poglavlja koja slijede.

- **'Srodna dijela'**

  U ovom poglavlju se opisuju postojeći objavljeni znanstveni radovi na temu detekcije vegetacije. Većina tih metoda dolazi iz područja daljinskih istraživanja gdje se detekcija vegetacije vrši na satelitskim snimkama. U ovom poglavlju se spominju i specifične upotrebe metoda za detekciju vegetacije i prilagodbe tih metoda za detekciju vegetacije u snimkama snimljenim na zemlji (ne-satelitske snimke).

  Napravljena je kategorizacija srodnih objavljenih metoda za detekciju vegetacija na osnovu dva kriterija:

  - Podjela po korištenim značajkama

    - *Metode bazirane na značajkama iz vidljivog dijela spektra*

      Postoje objavljene metode koje se baziraju na korištenju značajki iz vidljivog dijela spektra, ali s različitim primjenama. U nekim radovim za navigaciju autonomnih vozila autori su koristili dodatnu specijaliziranu opremu poput LiDARa koji služe za snimanje 3D okoline.

      Drugi radovi se razlikuju po odabiru korištenih značajki (različiti modeli boja i različite značajke za opis tekstura) za treniranje klasifikatora. Budući da je svaka od objavljenih metoda rađena za specifičnu primjenu (pojačanje boje u sportskim TV prijenosima, fotogrametrija, procjena rizika požara u područjima sušne vegetacije, i sl.) teško je napraviti usporedbu s metodama u ovom radu koje su također rađene sa specifičnom primjenom na umu.

    - *Metode bazirane na značajkama iz nevidljivog dijela spektra*

      U ovom području je puno više objavljenih radova koji se također razlikuju po primjenama i korištenim modalitetima u svrhu razvoja specifičnih značajki za detekciju vegetacije, tzv.vegetacijskim indeksima. Vegetacijski indeksi su opće prihvaćen način obrade satelitskih snimaka i u ovom poglavlju se može pronaći više detalja o samim svojstvima vegetacije u spektru elektromagnetskog zračenja i o raznim primjenama.

  - Podjela po pristupu razvoja modela

    - *Metode bazirane na strojnom učenju s ručno odabranim značajkama*

      Sve metode spomenute u prethodnim poglavljima su bazirane na tradicionalnom pristupu razvoja klasifikatora gdje se odabrane značajke koriste za treniranje odabranog modela. Ovdje je dan kratak pregled različitih objavljenih rješenja koja spadaju u ovu kategoriju.

    - *Metode bazirane na dubokom učenju s naučenim značajkama*

      Duboko učenje je još uvijek mlado područje istraživanja koje se sve više razvija što rezultira raznim treniranim modelima koji daju iznimne rezultate u svim područjima obrade slike od detekcije i klasifikacije pa do segmentacije.

  Model razvijen u sklopu ovog rada baziran na potpunim konvolucijkihm mrežama je detaljnije opisan u zasebnom poglavlju u nastavku.

- **'Metodologija'**

  U ovom poglavlju je sistematski opisano provedeno istraživanje.

  Budući da ne postoji javno dostupna baza koja bi se rimjenila u svrhu razvoja algoritama za detekciju vegetacije uz prometnice kao jedan od doprinosa ovog rada je baza koju smo snimili i ručno segmentirali. U potpoglavlju "Podatci" je opisana navedena baza te je detaljnije objašnjen postupak odabira podataka za treniranje. U svrhu predprocesiranja velikog skupa mogućih značajki za opis vegetacije korišten je *Wilcoxonov* test rangiranja značajki za izdvajanje najbolje rangiranih značajki za koje se pretpostavlja da najbolje (statistički značajno) opisuju razlike između željenih klasa (vegetacija i ne-vegetacija).

  Iduće potpoglavlje naziva "Alati za klasifikaciju" opisuje teoretsku pozadinu korištenih klasifikatora i korištene metode postprocesiranja. Najprije se uvode pojmovi i koncepti vezani za *Support Vector Machines (SVM)* koji je korišten za razvoj metoda strojnog učenja baziranih na ručno odabranim značajkama. Nakon toga se opisuju koncepti vezani uz razvoj neuronskih mreža, od predstavljenog matematičkog modela neurona pa do razvoja dubokih mreža.

  Razvijene metode za detekciju vegetacije su predstavljene u potpoglavlju "Metode klasifikacije" počevši od najjednostavnije, koja je nadograđivana u kompleksnija rješenja predstavljena u svakom idućem potpoglavlju. Svaka od predstavljenih metoda sadrži opisanu ideju koja je dovela do tog rješenja, implementacijske detalje metode, korištene alate i postignute rezultate zajedno s uočenim problemima te diskusijom o tome kako pristupiti njihovom rješavanju.

  Ukupno su predstavljene četiri metode:

  - ***Klasifikacija bazirana isključivo na značajkama boje***

    U postojećim objavljenim metodama za detekciju vegetacije autori kao značajke često koriste različite modele boja, dok neki iz modela izuzimaju komponentu osvjetljenja s idejom da trenirani klasifikator bude manje osjetljiv na promjene u osvjetljenju scene.

    Kao prvi korak istraživanja izvršena je analiza različitih modela boja za detekciju vegetacije. Kao značajke korišteni su RGB, HSV, YUV i CieLAB model boja te se sa svakim zasebno trenirao SVM klasifikator sa radijalnim kernelom. Također su testirani i modeli boja s izuzetom komponentom osvjetljenja.

    Rezultati su pokazali da korištenje HSV, YUV i CieLAB modela boja rezultira sličnom točnosti klasifikacije piksela, dok je RGB rezultirao nižom točnosti što nije iznenađujuće budući da je RGB aditivne prirode za razliku od ostalih modela. Korištenje svih komponenti modela je dalo bolje rezultate nego izuzimanjem komponente osvjetljenja iz vektora značajki.

U ovoj kategoriji su testirani i indeksi vegetacije koji su bazirani na značajkama iz vidljivog dijela spektra (VVI i GRVI). Rezultati postignuti s navedenim indeksima vegetacije su bili lošiji od rezultata dobivenih korištenjem nekog od modela boja i SVMa kao klasifikatora. Taj rezultat nije iznenađujući budući da su ovi indeksi razvijeni u području daljinskog istraživanja i namjenjeni su za primjenu na satelitskim snimkama. Primjena takvih značajki na snimke snimljene iz vozila očekivano nije urodila plodom.

Kao rezultat analize raznih značajki boja odabran je CieLAB model kao deskriptor boje, s kojim je postignuta točnost od $91,9015\%$, za razvoj kompleksnijih modela u nastavku.

Koristeći samo boju za treniranje klasifikatora je rezultiralo klasifikatorom koji objekte slične vegetaciji po boji pogrešno klasificira kao vegetaciju (npr.zeleni objekti). Da bi riješili taj problem potrebno je dodati dodatne značajke za opis teksture koja je, pored boje, najizraženija značajka vegetacije.

- ***Klasifikacija korištenjem značajki boje i jednostavnog deskriptora teksture***

  Prilikom odabira znčajki za treniranje klasifikatora koristimo vlastito znanje o problemu koji želimo riješiti. Ljudsko oko pored boje raspoznaje i teksture te na osnovu te dvije značajke ljudima nebi bio problem razaznati vegetaciju od ostatka sadržaja. U slikama, područja vegetacije su turbulentna zbog same prirode vegetacije (npr. vlati trave) koja raste u svim smjerovima te može biti i različitih oblika i veličina. U usporedbi s npr. glatkom površinom (zelenog) automobila područja vegetacije u slikama bi bila "kaotičnija".

  Entropija je mjera sadržaja informacije koja se može primjeniti na slike. Pretpostavka je da bi dijelovi slike koji sadrže više informacija imali veću entropiju od uniformnijih dijelova.

  Entropija se obično računa na crno-bijelim slikama, ali se isto tako može izračunati i za pojedine kanale različitih modela boja (npr. zašto ne zelena?). U svrhu pronalaska najbolje značajke entropije za razlikovanje vegetacije od ne-vegetacije upotrijebljen je statistički *Wilocxonov* test kao korak predprocesiranja značajki. Vrijednost entropije je izračunata za sve kanale (R, G, B, H, S, V, L, A, B Y, U, V i crno-bijeli) za obje klase (vegetacije i ne-vegetacija) i primjenjen je spomenuti test. Ideja je da su značajke sa višim rangom statistički značajnije za diskriminaciju klasa. Na osnovu testa kao značajku teksture odabrali smo entropiju izračunati za kanal S u prozoru o $9x9$ piksela.

Primjena entropije kao dodatne značajke je poboljšala rezultate točnosti klasifikacije na razini piksela (92,436%), ali problemi sa tzv 'problematičnim' slikama nisu riješeni u potpunosti. Dijelovi zelenog automobila su točno klasificirani kao nevegetacija, ali neki dijelovi koji su sadržavali više informacija (npr. rubovi) i samim time rezultirali višim vrijednostima entropije su doveli do pogrešne klasifikacije.

- ***Klasifikacija korištenjem značajki boje i kompleksnijeg deskriptora teksture baziranog na kontiuniranoj valićnoj transformaciji***

  Analiza problema iz prošlog poglavlja dovodi do zaključka da je potrebna kompleksnija reprezentacija značajke teksture. Uzimajući u obzir prirodu snimanja i snimljenih podataka ideja je bila naći značajku teksture koja bi bila neovisna o skali i rotaciji.

  Neovisnost o skali je bitna jer se snimanjem vegetacije uz prometnice snima vegetacija na različitimm udaljenostima od kamere, a vegetacije udaljenija od kamere je manje u fokusu što bi trebalo 'uhvatiti' nekom značajkom s kojom se može korigrati skala.

  Neovisnost na rotaciju je bitna jer se isti tip vegetacije može naći pod raznim kutovima (npr.vlati trave).

  Uzimajući u obzir sve navedene preduvjete, odabrana je kontinuirana valićna transformacija (eng. *Continuous Wavelet Transform - CWT*) za računanje značajke teksture. Kontinuirana valićna transformacija je opisana filtriranjem ulazne slike s familijom valićnih funkcija koja je konstruirana tako da se odabrani bazni valić dilatira (razne skale) i rotira što rezultira skupinom filtera pomoću kojih se detektiraju različite vrste uzoraka u slici.

  Za računanje valićne transformacije potrebno je definirati 5 parametara kojima se definira izgled osnovnog valića te sve skale i rotacije za koje se želi kreirati filter. Da bi dobili najoptimalniji skup parametara korišten je *Wilcoxonov* test rangiranja značajki pomoću kojeg su se isfiltrirale najoptimalnije vrijednosti.

  Koristeći novi skup značajki treniran je SVM klasifikator. Analiziranjem problematičnih slika (npr. slika sa zelenim automobilom) vidljivo je poboljšanje u usporedbi s prethodnim rezultatima čak i sveukupnoj točnosti klasifikacije piksela - 93,89%. Veći dio problematičnih objekata je sad točno klasificiran, no ne u potpunosti. I dalje ostaju problematični dijelovi slike koji sadrže visoke frekvencije poput rubova.

  Navedena metoda je dalje nadograđena s dodatnim predprocesiranjem u kojem se pomoću specijalne implementacije optičkog toga odredi područje interesa definirano kao područje u fokusu, tj.područje bliže vozilu. Detekcija se vrši samo unutar područja interesa - na ovaj način se smanji broj piksela koje treba klasificirati i isključiti potencijane piksele koji su predaleko od kamere i čija kvaliteta (razina detalja) može varirati.

- ***Klasifikacija bazirana na dubokom učenju koristeći potpuno konvolucijke neuronske mreže***

  Prethodne metode su bazirane na ručnom odabiru značajki korištenih za treniranje klasifikatora. Duboko učenje je specijalno po tome što se mrežama daju čisti podaci na ulazu, za razliku od prethodnih pristupa gdje se na ulazu davao vektor izračunatih značajki. Mreža se zatim trenira i u procesu treniranja skriveni slojevi mreže uče raspoznavati različite uzorke na osnovu kojih će cjelokupna mreža naučiti značajke koje će na izlazu dati rezultat što sličniji očekivanom (pomoću označenih, ručno segmentiranih ulaznih podataka).

  Korišteni model mreže je modificirani VGG16 model za klasifikaciju slika. Modifikacija mreže se sastoji u pretvorbi svih slojeva u konvolucijske slojeve i dodavanjem tzv. 'skip' koraka u mreži koji služe za povećanje rezolucije slike unutar mreže u različitim fazama učenja što nam omogućuje da se na izlazu dobije segmentirana slika jednakih dimenzija kao i na ulazu u mrežu.

  Izlaz iz mreže je segmentirana slika gdje svaki piksel predstavlja vjerojatnost da taj piksel pripada klasi 'vegetacija'. Postavljanjem praga koji se izračuna na osnovu crtanja ROC krivulje se kao konačni izlaz dobije binarna slika s označene dvije očekivane klase.

  Ova metoda je rezultirala s najvišom točnosti klasifikacije piksela - 96.257%

Uspordeba rezultata iz ovog rada s drugim metodama segmentacije i klasifikacije je teška budući da se način snimanja, korištena baza slika i planirana primjena značajno razlikuju od trenutno objavljenih metoda te bi takve rezultate bilo teško tumačiti. Detekcija i raspoznavanje objekata u video sekvencama je relativno istraženo područje, dok raspoznavanje razlčlitih vrsta površina u prirodnim okruženjima je manje istraženo područje gdje spada i detekcija vegetacije. Trenutno ne postoje srodni radovi koji istražuju područje detekcije vegetacije primjenom značajki iz vidljivog dijela spektra.

U Tablici 1 su predstavljeni svi rezultati dobiveni kroz ovaj rad (bez postprocesiranja - rezultati treniranih klasifikatora). Najprije su uspoređeni različiti modeli boja sa i bez komponente osvjetljenja gdje je CieLAB dao najbolju točnost i kasnije je korišten u kombinaciji s različitim značajkama teksture (Entropija i CWT). Evaluirane su i performanse vegetacijskih indeksa baziranih na R, G i B vrijednsotima koje pripadaju značajkama vidljivog dijela spektra.

Konvolucijske mreže daju najbolje rezultate i to ne samo u postotku točno klasificiranih piksela, nego i u točnosti klasifikacije problematičnih objekata što je bio i jedan od glavnih problema koji smo prethodno pokušavali riješiti kompleksnijim značajkama teksture.

Table 1: Pregled točnosti klasifikacije metoda za detekciju vegetacije.

Vektor značajki	**Točnost**	**Vektor značajki**	**Točnost**
CieLAB	91,9015%	AB	89,9947%
YUV	91,8235%	UV	61,7462%
HSV	91,4285%	HS	88,7151%
RGB	89,6314%		
VVI	58,342%		
GRVI	67,617%		
CieLAB + Entropija	92,436%		
CieLAB + CWT	93,89%		
FCN	96,257%		

- **’Zaključak’**
  Posljednje poglavlje sažima sve zaključke i rezultate dobivene provedenim istraživanjem. Detekcija vegetacije može poslužiti kao prvi korak za razvoj algoritama klasifikacije tipova vegetacije što bi imalo široku primjenu u ekologiji i botanici za mapiranje staništa, istraživanje šumovitih, teško-prohodnih područja, za praćenje stanja i zdravlja vegetacije i sl.

Ključne riječi: Obrada slike, detekcija vegetacije, održavanje prometnica, strojno učenje, duboko učenje, konvolucijske neuronske mreže

Contents

Chapter 1

Introduction

1.1 Motivation

Vegetation detection is a topic frequently addressed in remote sensing, but recently it has been gaining importance in the field of robotics. The need for detecting vegetation in robotics arose when autonomous vehicles (AVs) started being used for forest exploration. Detection and analysis of vegetation from satellite images is used to monitor the health of vegetation which is affected by urbanization and industrialization. Detection of stressed vegetation [1, 2] is aimed at raising environmental awareness and improving our ecological footprint. Aerial vegetation detection is useful in forest management planning, urban vegetation mapping, environmental monitoring [3], vegetation cover estimation [4], estimating land usage in urban areas [5], mapping arctic vegetation [6], phenological surveys [7] and in drought-affected vegetation detection which is used for fire risk assessment [8, 9].

The increase in demand and popularity of autonomous vehicles resulted in increased research of vegetation detection as an obstacle in off-road navigation. First works on this topic were greatly influenced by The DARPA Grand Challenge for developing a self-driving car. Vegetation is considered a soft obstacle which an off-road vehicle can drive over compared to hard obstacles which must be avoided [10, 11, 12, 13, 14, 15].

Vegetation has very distinctive properties in the Near Infra Red (NIR) part of the spectrum because the chlorophyll in vegetation significantly reflects NIR light. That property is used in many published methods for vegetation detection. Our research on detecting vegetation was focused on using only the information from the visible spectrum to allow the usage of a common color camera. The benefit of this approach is that the same camera can be used for other computer vision tasks which require color images as input data.

In the following chapters, we provide a broad review of the vegetation detection research area with an emphasis on a specific application of vegetation detection methods that are based on some specific constraints.

We will present several methods that we have developed, ranging from simpler ones to more complex methods used for roadside vegetation detection. The planned application was intended for traffic safety and infrastructure maintenance (e.g. mowing the grass along the road). Roadside vegetation maintenance is usually done by a maintenance worker who manually operates a mower, trims what is necessary and avoids posts, trees, and other obstacles that would possibly damage the mower. Mounting a camera on a maintenance vehicle and using a machine learning-based vegetation detection algorithm to navigate the mower would greatly help workers who maintain traffic roads.

The main goal is to demonstrate that different machine learning approaches can be effectively used to detect roadside vegetation in color images without using any special equipment. Efficient detection of vegetation from a camera mounted on a vehicle provides many possibilities for this application even beyond traffic infrastructure maintenance which will also be discussed.

1.2 Dissertation structure

The thesis is organized as follows: a brief overview of different proposed methods for vegetation detection is given in Chapter 2. The research path of the doctoral research consisted of applying different approaches (methods and algorithms) which were used to improve and refine the vegetation detection method. This whole process is described in Chapter 3, accompanied by results and discussion. The final conclusion and discussion are presented in Chapter 4.

Chapter 2

Related Works

When developing a detection or classification algorithm based on machine learning one faces two main problems:

- What to use for training?

  We need a good dataset in order to properly train and test the selected classifier. Generally, the better the data, the better the classifier/detector.

  Depending on the selected machine learning model the data for training can be the original (unchanged) data or a set of calculated features that describe the original data. Finding a good feature set is a neverending task. The more (good) features you use, you can achieve a better classifier, but overtraining a classifier can also be a problem. That is why this step is so important and can be very time-consuming.

- What to train?

  Once you have your data or you have selected a feature set that best describes your data you choose a model to train.

This process (construct a feature set -> use the feature set to train a classifier) is common in a machine learning-based approach. The feature selection process is very sensitive and it depends on the data and on the problem that is trying to be solved. We will encounter some of these issues in later Chapters.

On the other hand, in a Deep Learning-based (DL) approach one is freed from the first step of selecting features for training the model. One of the biggest benefits of choosing a Deep Learning model is its ability to perform automatic feature extraction from raw data, also called feature learning. Deep Learning algorithms seek to exploit the unknown structure in the input distribution in order to discover good representations, often at multiple levels, with higher-level learned features defined in terms of lower-level features [16].

2.1 Detection based on input data

The human visual system detects vegetation on the basis of typical vegetation's visual characteristics such as color, texture, and shape. Hence, one plausible research direction is to mimic the human visual system when developing methods for automatic detection of vegetation. Besides typical features in the visible spectrum, studies have shown that there is also valuable information for vegetation detection in the spectrum invisible to the human eye and therefore, methods for detecting vegetation can be divided into two groups:

- Methods based on the visible spectrum
- Methods based on the invisible spectrum

2.1.1 Methods based on the visible spectrum

Methods based on the visible spectrum use color and/or texture features for detection [17]. Color features alone are not sufficient because objects similar in color to vegetation can be mistaken for vegetation. Such objects can be distinguished from vegetation by additional texture features. Some researchers use LiDAR (Light Detection And Ranging) sensors that measure the distance to a target by illuminating that target with laser light. This provides 3D information about the environment and can be used for calculating spatial texture features. Authors in [13, 14] use a sliding cube across the 3D-point cloud and segment the 3D-point cloud into three classes: surfaces, linear structures and porous volumes (foliage grass, tree canopy). It can be argued that using only 3D-data, i.e., only texture properties, cannot result in robust detection of vegetation because no color information is used. Therefore, in [13] the authors combine 2D and 3D information for vegetation detection. The problem with this approach is mapping 2D and 3D data because they are obtained by two different sensors. This approach is time-consuming and it should be used when time is not critical [14]. In a similar way authors in [11] use a combination of 2D and 3D features. They use the height calculated from the 3D LiDAR data and H and S color components (from HSV color space) for color features.

In addition to its use in navigation systems, which is the most common, vegetation detection is also used in improving the quality of video or TV images, especially in sports broadcasts. In [17, 18] authors used YUV color components and texture features for vegetation detection. Based on detected vegetation regions, the image is enhanced by changing the color, brightness or saturation of pixels.

There are also so-called vegetation indices that use only information from the visible spectrum, combining color channels to obtain a single feature. For example, the Visible Vegetation Index (VVI) is a measure of the amount of greenness of an image [19], while in [20] authors evaluate the use of the Green-Red Vegetation Index (GRVI) as a phenological indicator.

Another example where vegetation detection is applied is in photogrammetry which, for example, uses vegetation detection to monitor landslides and to eliminate the influence of vegetation in the process of monitoring land deformations [7]. To achieve this, a Bayesian classifier was trained using texture features obtained from the gray-level co-occurrence matrix in conjunction with Sobel gradient values.

One example of using vegetation detection similar to ours can be found in [8, 9]. The emphasis of their work was to develop a novel strategy for feature extraction used for roadside grass classification [8]. In [9] the authors focused on developing a novel texture feature based on a multiple classifier technique with the goal of improving the classification of dense and sparse grass areas. The goal was to use the detected areas for fire risk assessment. Another vision-based approach for roadside vegetation detection is presented in [21]. The authors constructed a superpixel database by segmenting training images into superpixels, and each superpixel patch is represented with multiple features. The detection is performed by superpixel matching.

2.1.2 Methods based on the invisible spectrum

Using features from the visible spectrum has some drawbacks (leaves tend to change color, the presence of green objects, etc.) and that is why features from the invisible spectrum are used more often. This idea came from remote sensing where researchers found that chlorophyll-rich vegetation has a high reflectance of NIR wavelengths, while it strongly absorbs red light during photosynthesis [22]. Using spectral band information researchers have developed various Vegetation Indices which are a mathematical combination of different reflectance values that emphasizes the spectral properties of vegetation so that it appears distinct from other image features.

When electromagnetic radiation hits an object, three different interactions can occur:

- it can be *transmitted* through the object like light through a window
- it can be *reflected* off the object like light bouncing off a mirror
- it can be *absorbed* by the object.

The reflected radiation is the reason we, with our eyes, see different objects and discern different colors. Different colors depend on the combinations of different wavelengths that are reflected from the object. They make the so-called visible part of the electromagnetic radiation spectrum. The value of reflectance ranges from 0 (no reflection) to 1 (complete reflection).

In Fig. 2.1 three spectral curves are shown for three different objects (vegetation, water and ground) [23]. A spectral curve or the spectral signature of the object is obtained by plotting the reflection values associated with that object for a relevant wavelength range.

The vegetation reflection spectrum is divided into three areas:

- visible (from $0.4\mu m$ to $0.7\mu m$)
- near infrared (NIR) (from $0.701\mu m$ to $1,3\mu m$)
- middle infrared spectrum (MIR) (from $1,301\mu m$ to $2,5\mu m$)

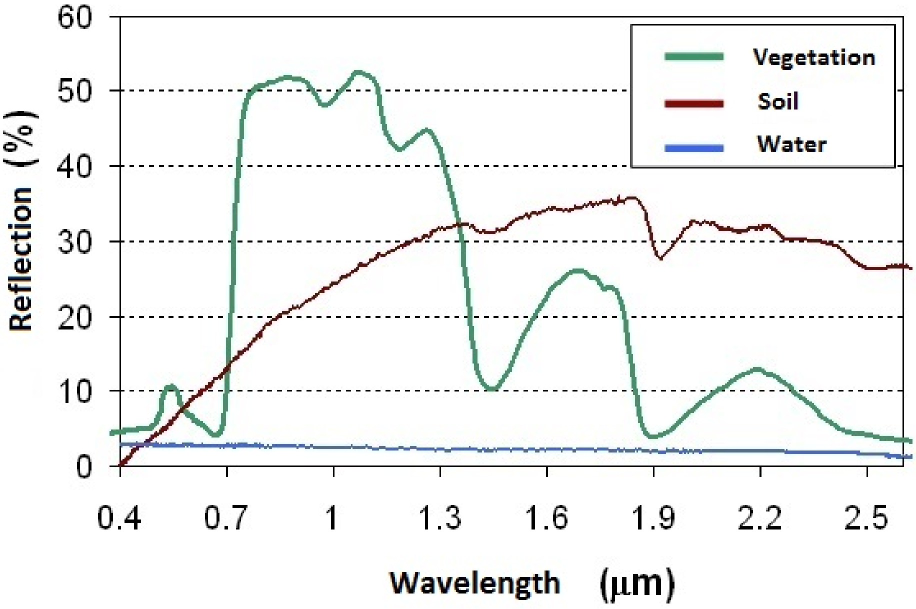


Figure 2.1: Spectral curve for three different types of objects

as shown in Fig. 2.2.

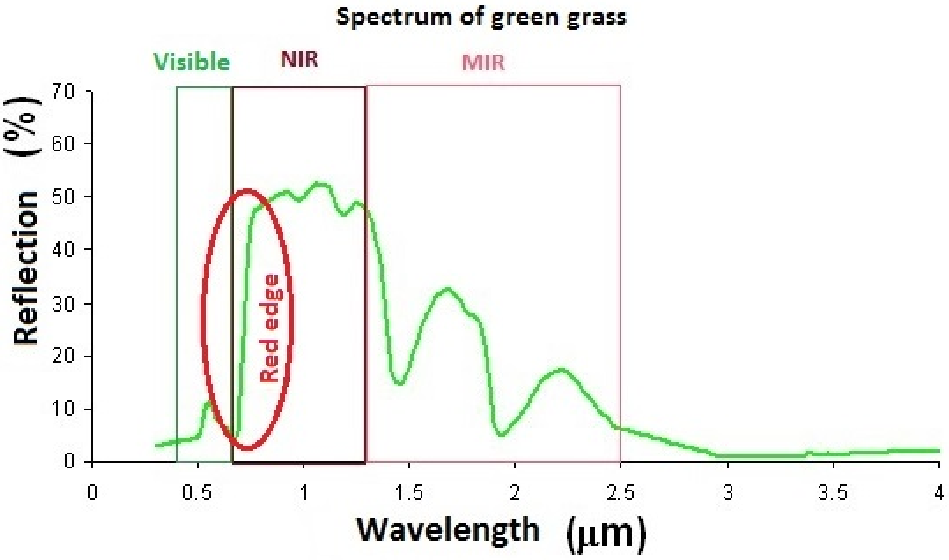


Figure 2.2: Spectral curve of green grass

In Fig. 2.2 the so-called ’red edge’ is visible that represents a sudden jump in the spectrum between the large absorption of wavelengths of visible red and the strong reflections of NIR wavelengths. In the visible part of the spectrum, the chlorophyll found in vegetation absorbs red light and reflects green (hence, the green color). The amount of reflection in the NIR area is very pronounced and this characteristic is suitable for vegetation detection.

Many Vegetation Indices have been developed during the last two decades. Depending on the spectral properties of plants different spectral ranges are used for calculating these indices.

It can be observed that many scientists have developed indexes related to their specific field of research. One typical representative, which is most commonly used, is NDVI (Normalized Difference Vegetation Index) [3, 4, 12]:

$$NDVI = \frac{NIR - Red}{NIR + Red} \tag{2.1}$$

where NIR and *Red* are the spectral reflectance measurements acquired in the NIR and *Red* (visible) regions, respectively.

NDVI values can range from -1 which represents a blue sky to 1 which indicates chlorophyll rich vegetation. The reflection relationship in the red and NIR area is shown in Fig. 2.3 [10].

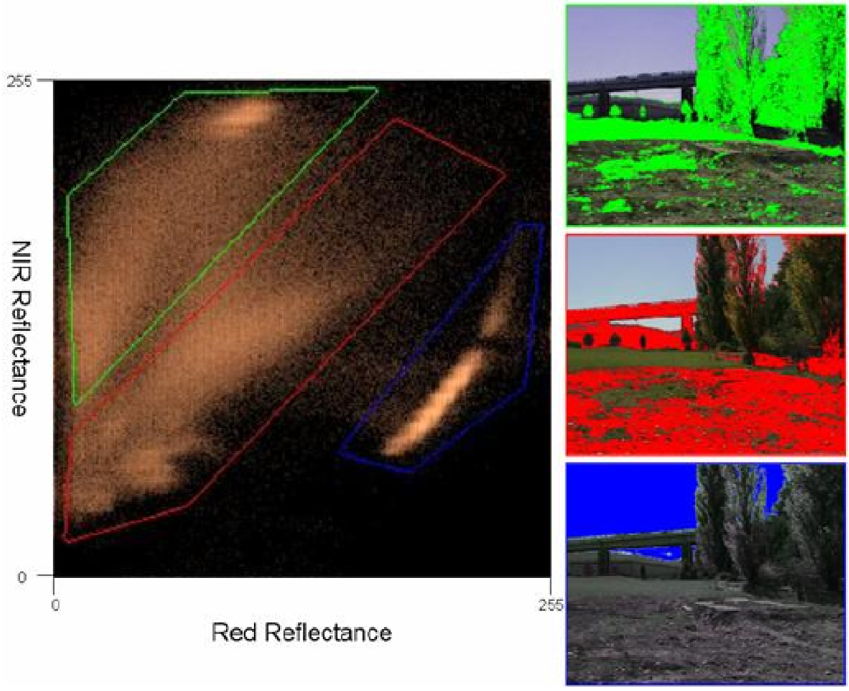


Figure 2.3: Ratio of Red and NIR reflectance

Although NDVI has been successfully used in remote sensing, the work of Bradley et al. [10] has shown that there is a drastic difference between the viewpoints of a satellite and of an autonomous ground vehicle since there are more problems typical for on-ground recordings such as shadow, light reflection, and underexposure effects.

Taking that into consideration and wanting to avoid time-consuming processes of calibration and LiDAR scanning, Nguyen et al. developed a modified vegetation index MNDVI [14] which they use as a feature for vegetation detection:

$$MNDVI = \frac{NIR - log(Red + \varepsilon)}{NIR + log(Red + \varepsilon)} \tag{2.2}$$

where $\varepsilon(\leq 1)$ is a constant used to avoid a negative index value and the logarithmic term expresses the less impact of the Red when an artificial lighting system is used.

In [12] they combine NDVI and MNDVI to obtain pixels rich in chlorophyll. These pixels are seeds for the spreading algorithm that follows. For each seed pixel, the distance in color and texture between the seed pixel and its neighbors is calculated and based on those distances it is decided if the neighboring pixel belongs to the same group as the seed pixel or not.

Systems based on information from the invisible spectrum require equipment for recording NIR images and systems based on the visible spectrum that uses LiDAR for feature extraction need an additional LiDAR scanner. Since our focus is on using only features from the visible part of the spectrum all we need is a color camera for data acquisition and no additional specialized sensors are required.

2.2 Detection based on Machine Learning

There are many different applications of vegetation detection techniques and they all vary in types of features used and classifiers trained. In order to achieve good results with conventional Machine Learning (ML) techniques, the best features have to be selected. Deep Neural Networks features are learned from the input data and that is the main difference in these two approaches.

We will, first, present some of our methods and results achieved by manually selecting features and training a learning model. Through these experiments, we will present all the problems and setbacks encountered and in the end, we will present the result of using a Deep Learning approach where the features will be learned and not selected.

2.2.1 ML based on selected features

Most of the methods mentioned in Chapter 2.1 follow this approach when developing a detection and/or classification algorithm. The first step consists of selecting a set of features based on the domain knowledge that one has and using that carefully selected feature set for training an ML-based algorithm. The second step is to select an appropriate learning algorithm (e.g. Bayesian [7], SVM [11, 13], Linear maximum entropy [10]). Different algorithms have different advantages and disadvantages so some investigation should be done in order to select one.

The hardest and the most important part is selecting the feature set which will represent the dataset. Some statistical methods can be used in order to analyze possible features and select the subset that is believed to be the most significant one.

If a model is trained on bad data/features then we can not expect good results. Going back and changing the features is not rare. The biggest difference, when compared to Deep Learning methods, is the feature selection step. For DL models, the researcher does not handpick the features because the features are learned from the raw input data and the given labels corresponding to different classes in the images.

2.2.2 ML based on learned features

Such approaches have shown promising results in solving a variety of problems, such as object recognition in images [24, 25], speech recognition [26], semantic segmentation of images [27, 28, 29, 30, 31, 32], handwritten character classification recognition [33] and text analysis [34]. Prior approaches that used Convolutional Neural Networks (CNN) for semantic segmentation label each pixel with the class of its enclosing object or region. Fully Convolutional Networks [35] are a special derivative of CNNs that are trained end-to-end, pixel-to-pixel and exceed the state-of-the-art in many semantic segmentation problems. Using such an approach, the authors have built a fully convolutional network that takes input images of arbitrary size and produces a correspondingly-sized output with efficient inference and learning.

The work of [36] investigates the use of deep fully convolutional neural networks (DFCNN) for pixel-wise scene labeling of satellite images using data fusion from heterogeneous sensors (optical and laser). The approach is able to generate dense scene labels using multi-modal and multi-scale encoder-decoder architecture. Another, similar approach for dense pixel-wise segmentation based on multi-channel data fusion is presented in [37]. The authors train their architecture with RGB, NIR, and depth data contained in their database. Then they identify the best performing modalities and fuse them by using a DFCNN.

These papers focus mostly on multi-modal data fusion. There has also been work done in the area of vegetation classification represented in [38] where the main goal is to distinguish weeds from crops. The data used was recorded with a multispectral camera. Here, the authors use a pipeline of two different CNNs applied to the input RGB and NIR images. The first step is the detection of 3D points that belong to green vegetation. Using the NIR values in this task is highly useful which has been shown even in hand crafted-based approaches mentioned before. CNN can take advantage of such a feature even better. The second, deeper, CNN is then used to classify the extracted pixels into desired classes.

Recently, deep learning was used to create a new NDVI-based index. The authors in [39] use a Conditional Generative Adversarial Network (CGAN) architecture model to generate a new vegetation index.

Deep learning is still emerging as a tool in many applications and lately, it has been researched and used with multi-spectral imagery for various tasks. Since our focus is on detecting vegetation by using data from the visible spectrum, a logical continuation of our work is to investigate the usage of a deep learning approach for this task. Currently, there is no research done on this specific topic.

Chapter 3

Methodology

In this chapter, four different classification methods for detecting roadside vegetation will be presented starting from the simplest approach where only color features were used as descriptors, followed by variants of manually selected color and texture features used for training a classifier, and finishing with a Deep Learning method where no features were manually selected:

- Methods with manually selected features
    - Only color features
    - Color and entropy-based texture features used with an SVM classifier
    - Color and CWT-based texture features used with an SVM classifier
- Method with learned features
    - Deep learning-based method using Fully Convolutional Networks

Going through the previous chapter and analyzing related methods developed for vegetation detection one can notice that most of the methods follow the recipe:

- choose features (most often color and texture features)
- use those features and machine learning to train a classifier
- test and refine the classifier

There are many possible features and machine learning methods to choose from and precisely those make the difference between different published methods and the results achieved.

Vegetation detection is a problem that can be tackled in many different ways. If the detection is done on the ground (from an AV vehicle) the selected features and/or classifier depend on the application, method of recording, type of recording, the application of the detection/classification method, and additionally, once a classifier is chosen there are parameters to fine tune for the problem at hand.

The idea was to try simpler methods and gradually get to more complex solutions in order to properly analyze the issue at hand and to identify the most beneficial components used.

3.1 Data

To the best of our knowledge, there is no publicly available database that could be used for our application. That is why we made our own database for testing. The database was recorded using an HD Camcorder Canon XF100 from a moving vehicle. There are several scenarios that we took into consideration when recording and selecting images that will represent our dataset used for verification. We wanted to create a versatile database which would contain as much as possible real-world scenarios. Some examples of images are shown in Fig. 3.1 where we covered:

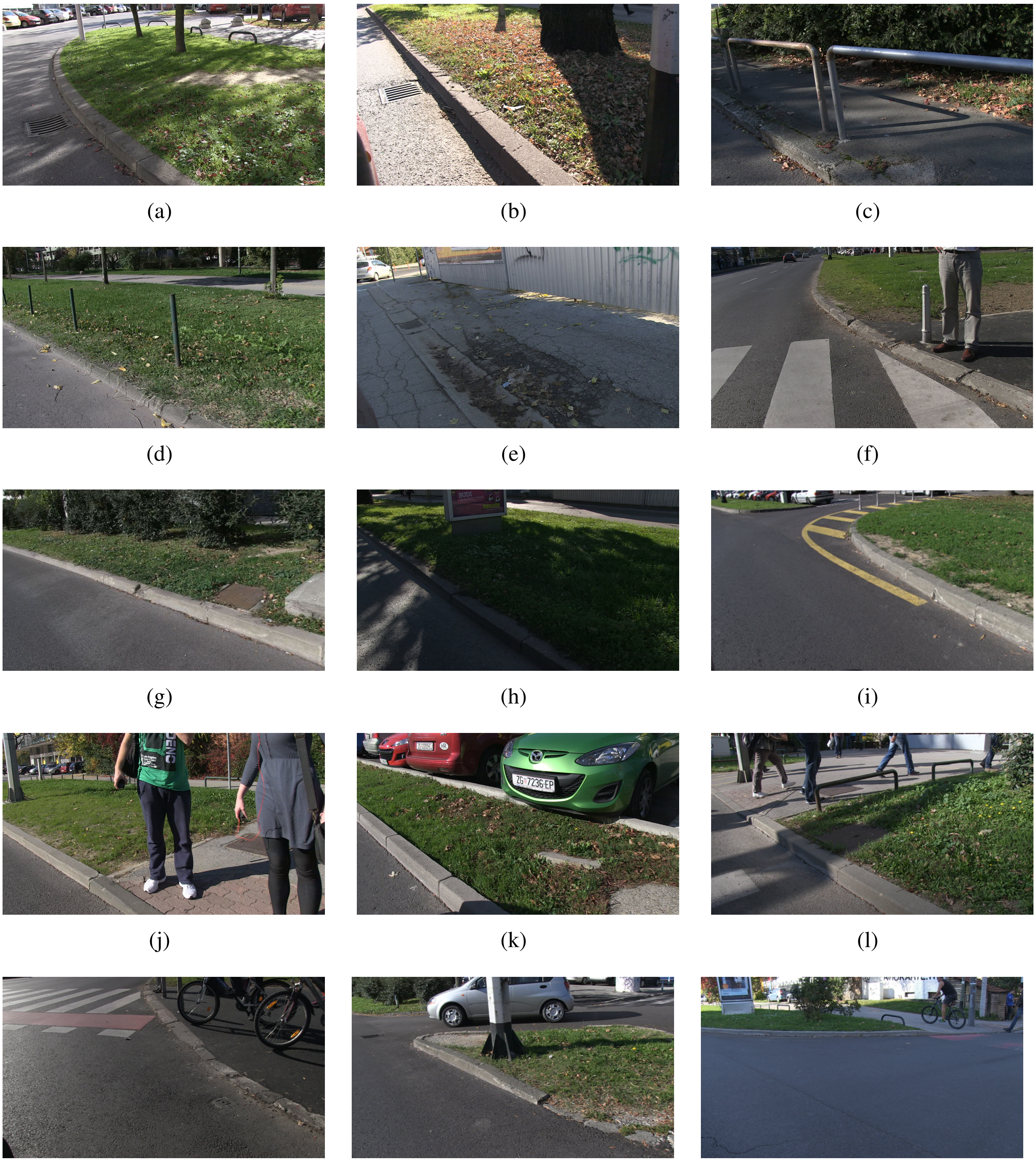
- Varying lighting conditions - The recording was always done in daytime but during different times of the day.
- Different traffic scenarios - Driving on main and various auxiliary roads, parking lots, and other traffic scenes that include special road signalizations and roadside infrastructure.
- Different types of roadside vegetation - Roadside vegetation mostly consist of grass, but often there are different types of bushes planted by the road, especially in urban environments.
- Vegetation of different colors - When seasons change, the leaves from trees turn brown/yellow and fall to the ground. These examples were included in the final dataset since it could also be used for roadside maintenance.

During training and validation of our algorithms, the database was carefully split into sets for training and validation keeping in mind that all scenarios and special cases are covered in both sets.

Currently, our database contains 270 images of resolution 1920x1080 extracted from the recordings. For every image, there is a corresponding label image used as ground truth for training and testing. The segmentation of label images was done in a pipeline which consisted of performing *k-means* clustering on the original image and then manually correcting the clustering result.

K-means clustering is one of the simplest unsupervised machine learning algorithms. The algorithm partitions the data into *k* mutually exclusive clusters. Each observation, i.e. pixel in an image, is treated as an object having a location in space and the goal of the algorithm is to assign each observation to its appropriate cluster. The decision to which cluster a pixel belongs to is made based on a defined distance metric [40].

The algorithm starts by randomly selecting *k* centroids, which are used as the beginning points for every cluster. The user defines the number of clusters *k* which is 2 in our case. The distance to the cluster centers is calculated for every observation and based on the calculated



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

(j) (k) (l)

(m) (n) (o)

Figure 3.1: Example images from the database: (a) Vegetation in shade; (b) Main road with roadside vegetation in bright sunlight and in shade with autumn leaves; (c) Auxilliary road with different bushes by the road; (d) Main road with roadside vegetation and green poles; (e) Auxiliary road without roadside vegetation; (f) Roadside traffic scenario with a pedestrian crossing; (g) Different type of roadside vegetation; (h) Roadside vegetation in heavy shade; (i) Traffic scenario with special yellow traffic markings on the road; (j) Traffic scenario with a pedestrian in a green T-shirt; (k) Traffic scenario with a parked green vehicle; (l) Traffic scenario with multiple pedestrians; (m) Traffic scenario with a pedestrian crossing withpout roadside vegetation; (n) Traffic scenario with a vehicle in motion and some roadside vegetation; (o) Roadside vegetation further from the camera on a traffic crossing

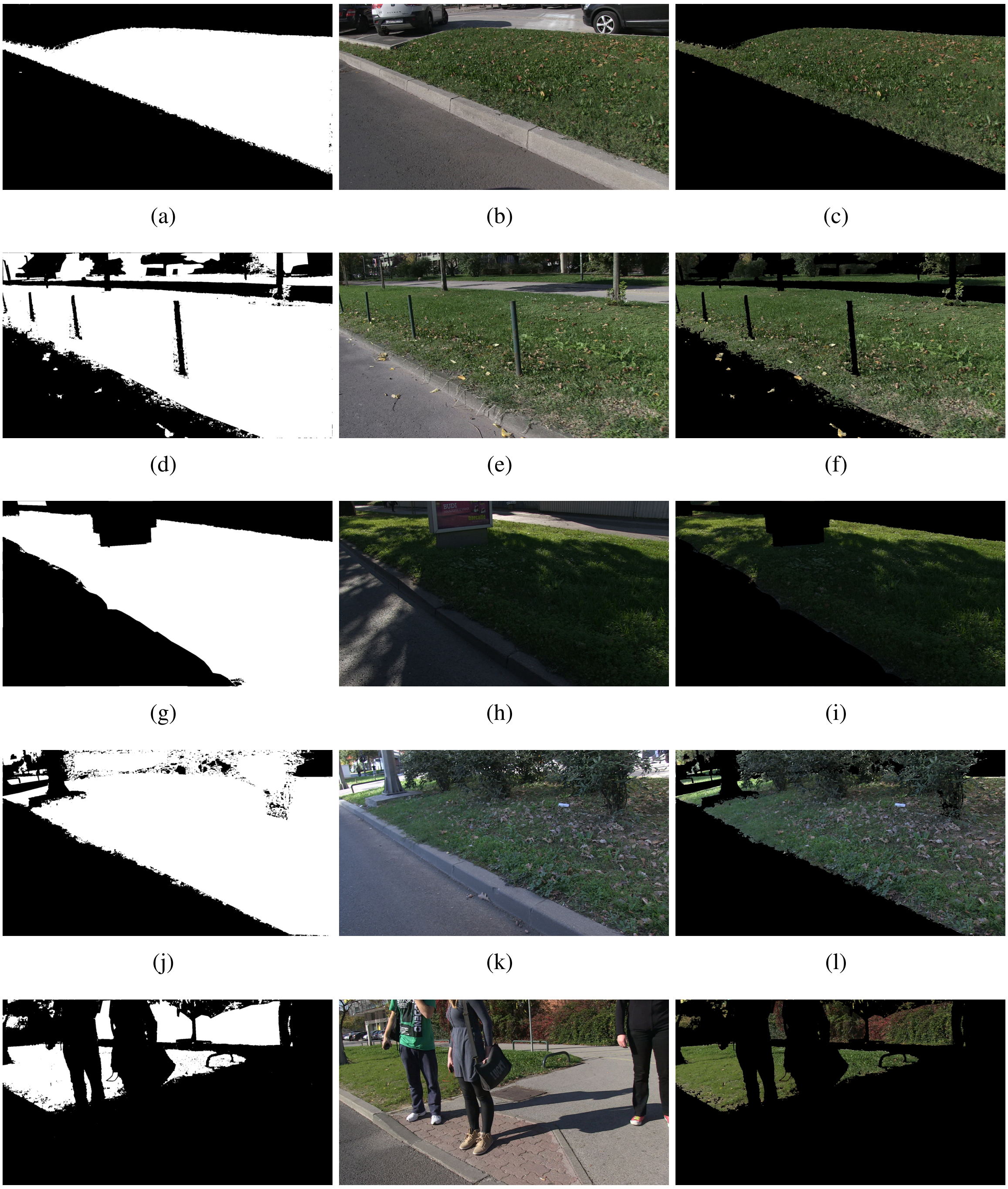
values, the object is assigned to one of *k* defined clusters (a smaller distance indicates higher similarity to the cluster representative). The algorithm iterates through the data and optimizes the positions of the centroids. Once the centroids have stabilized, i.e. there are no changes in their values, or if a predefined number of iterations has been achieved the algorithm stops.

The main issue with this algorithm is that if you run it several times on the same data you can get different results each time because the algorithm starts at a random location every time. It is also very sensitive to outliers since they can skew cluster grouping and increase the amount of time needed to find an optimal solution. All of that being said makes this algorithm not reliable to be used as a vegetation detection method, but it can help speed up the segmentation process. The biggest benefit was when segmenting images where vegetation was heavily present and the clustering often managed to group vegetation areas in one cluster which we then manually modified, if needed, to create a final label. Other images, where vegetation is not that present or where it was too scattered, k-means clustering often failed and did not manage to cluster all vegetation areas in one cluster so there was more work to be done manually. In Fig. 3.2 we show some images from the database together with their labels.

In these binary masks, ones ('1') represent the positive class, i.e. the vegetation areas, and the zeros ('0') represent the negative class, i.e. the non-vegetation areas.

Depending on the image content the difficulty of the labeling process varied drastically. For images like Fig. 3.2(b) containing simple scenarios where roadside vegetation is uniformly illuminated and contained in one area, the k-means algorithm often grouped the vegetation as one cluster which was then taken as the final image label. More often this was not the case. Images often contain some objects which would skew the k-means clustering of vegetation. In Fig. 3.2(e), for example, we had to manually correct all the green poles while in Fig. 3.2(n) we had even more manual labeling for selecting all the vegetation in the background and correctly labeling objects similar to vegetation in color.
K-means clustering as a pre-labeling step did speed up the labeling in scenarios where different types of vegetation were present in the image like in Fig. 3.2(k). In cases like this, the clustering helped in defining boundaries of vegetation areas, while minor corrections had to be done to, for example, also include the dry vegetation or vegetation in heavy shade like in Fig. 3.2(h).

It should be made clear that many publications mentioning successful vegetation detection often indicate some very specific species of vegetation or a very limited dataset. And these approaches are then just used for robots operating in a specific environment or for satellite images. They are not used for roadside vegetation detection using a camera mounted on a vehicle. Due to different goals, those methods cannot be directly compared to our method for detection of roadside vegetation.



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

(j) (k) (l)

(m) (n) (o)

Figure 3.2: Example of manually segmented label images: (a) Final label for a simple roadside vegetation scenario; (b) Original image containing simple roadside vegetation scenario; (c) Original image multiplied with its corresponding label; (d) Final label for image containing roadside vegetation and poles (e) Original image containing roadside vegetation and green poles; (f) Original image multiplied with its corresponding label; (g) Final label for image with vegetation in heavy shade; (h) Roadside vegetation in heavy shade; (i) Original image multiplied with its corresponding label; (j) Final label for roadside scenario with different types of roadside vegetation; (k) Different type of roadside vegetation; (l) Original image multiplied with its corresponding label; (m) Final label for a roadside traffic scenario with a pedestrian in a green T-shirt; (n) Traffic scenario with a pedestrian in a green T-shirt; (o) Original image multiplied with its corresponding label

3.1.1 Selecting the training set

Size of the training set

Since we are focused on per-pixel classification this means that there are 1920*1080*270 pixels available for training and testing the classifier. Training a classifier with a large number of samples can result in a complex model that can be computationally demanding. Smaller sample size needs to be selected in order to optimize the whole process.

It is possible to construct a formula that suggests the optimal sample size [41]. There are various statistics-based recipes for calculating the required sample size which requires knowledge of the variance or proportion in the population, the maximum desirable error, as well as the acceptable Type I error risk (confidence level).

Given a population size (1920*1080*270), a specific margin of error ($ME = 2.5\%$) and a desired confidence interval ($CI = 95\%$), based on the table from [41] we found that the best sample size is 4000 (2000 samples representing vegetation and 2000 samples representing non-vegetation). These 4000 samples are chosen randomly. We tested our algorithms by using a greater number of samples but it only increased the computational time and it did not significantly improve accuracy.

Class representation in the training set

Once we established the size of our training set we had to decide which samples are going in that limited set. Selecting the training data is an important step because all further classification depends on the trained model.

First, we need to pay attention to stratification. Stratification is a system of dividing the population into homogeneous subgroups/classes before sampling. Once these subgroups are defined, we then take samples from each individual group which ensures that every class is represented in the training dataset (e.g. 50% positive class samples and 50% negative class samples).

We have two classes: *vegetation* and *non-vegetation*. We could randomly select samples from these two classes, but an analysis of the database showed that there are corner cases and subgroups of samples we need to take into consideration.

The database analysis showed the following:

- **Not all vegetation is green** - using only green vegetation in training will make a classifier that does not detect dry vegetation or any other non-green vegetation. If samples of dry vegetation (or other non-green vegetation) are included in the training phase then the classifier will be more robust and it will learn to detect vegetation of different colors.
  - Vegetation does not only differ in color but also in shape and size. There are different types of roadside vegetation (grass, leaves, bushes, etc.) to be detected which will

have to be represented in the database and it will be something to pay attention to when selecting the training feature set.

- **Everything green is not vegetation** - the non-vegetation class is more complex since it contains everything except vegetation. In order to represent such a versatile group, we need to identify potentially problematic cases and be sure to include them.
  Through our research and testing, we identified some special cases (i.e. green and yellow vehicles, green roadside posts, yellow traffic markings on the road, etc.) which we labeled in images. These labels were used when selecting training samples to ensure that they will be represented in the training sample set.

3.1.2 Feature ranking

The biggest issue when choosing and training a classifier is to select the feature set to be used for training. If one knows that color is important for the task at hand there are several color modulations to choose from. For texture features, there are even more possibilities. How to choose the best one? To select the optimal (hopefully the best) features we use additional statistical tools to help us narrow down the selection.

To understand the influence of features on the system, or even if the number of features is too large, feature ranking is a good method to get baseline results, to assess features individually, and is independent of the choice of the predictor [42].

Feature ranking is a good filtering method and a good preprocessing step where features are ranked using some statistical test and based on the calculated rank we choose the best one as our descriptor. The goal of using any kind of filtering of features is to eliminate possible outliers.

In our work, we used the Wilcoxon rank sum test which is a nonparametric test for equality of population medians of two independent samples. A nonparametric test was used because they don't assume normally distributed classes. Our samples were tested and showed that they do not follow a normal distribution [43].

Variable ranking makes use of a scoring function computed from the input variables (calculated features) and output variables (class assignment). The Wilcoxon rank sum test uses the rank sum statistic calculated for every feature in the feature vector as the scoring function. By convention, we assume that a high score is indicative of a valuable variable [42]. By filtering our feature vector, according to the calculated rank, we eliminate less valuable features.

3.2 Classification tools

The following chapters contain more information about the specific implementations of used classifiers together with the achieved results, but first, we will present some theoretical background and introduce and define some concepts and methods that will be mentioned in the following chapters and which will serve to better understand the presented methods.

3.2.1 Classifiers

Support Vector Machines

Support Vector Machines [44] is a well known supervised learning algorithm used for both regression and classification tasks. SVM is used as a classifier in some of the methods that will be presented in this thesis.
The objective of this algorithm is to find a hyperplane that distinctly classifies data points into their respective classes. Depending on the dimensionality of the data the hyperplane can be a line (in a 2D feature space) or a plane (in a 3D feature space) as shown in Fig. 3.3.

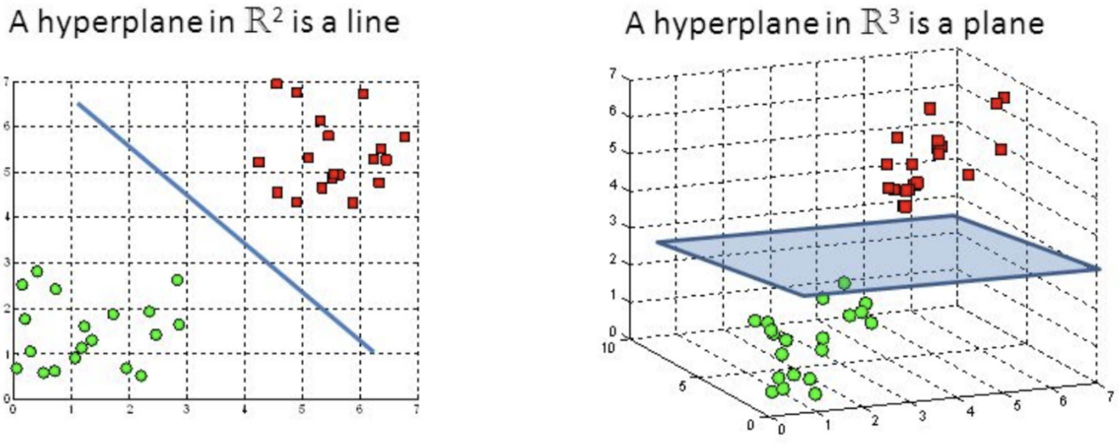
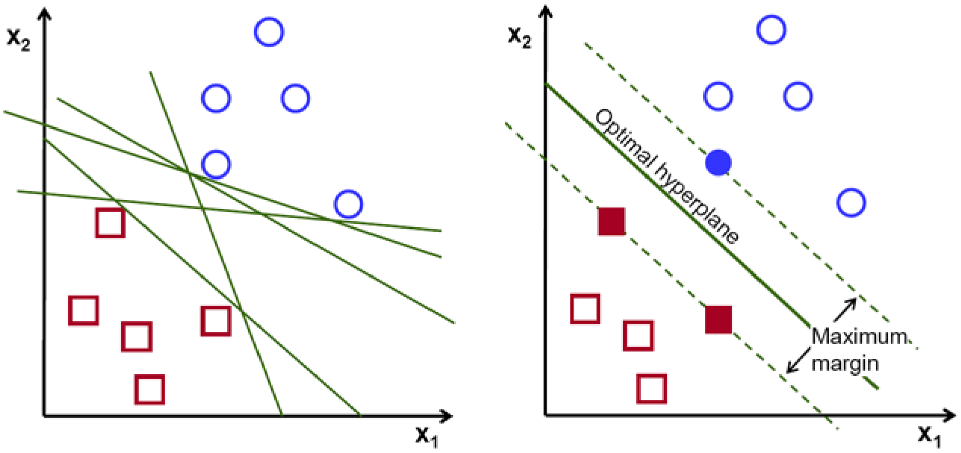


Figure 3.3: Hyperplanes in 2D and 3D feature space

There are many possible hyperplanes that could be chosen to separate two classes of data points. The goal of SVM is to find the *optimal* plane, i.e. the plane that has the maximum margin as shown in Fig. 3.4. Maximizing the margin distance provides some assurance that future data points will be classified with more confidence.

Support vectors (where the method gets its name) are data points that lie closest to the hyperplane and influence the position and orientation of the hyperplane. As shown in Fig. 3.5 depending on the chosen support vectors the margin of the classifier will change. The SVM algorithm is based on maximizing this margin around the separating hyperplane. The problem



(a) (b)

Figure 3.4: (a) Possible hyperplanes (b) Optimal hyperplane

of finding the optimal hyperplane is an optimization problem solved by different optimization techniques.

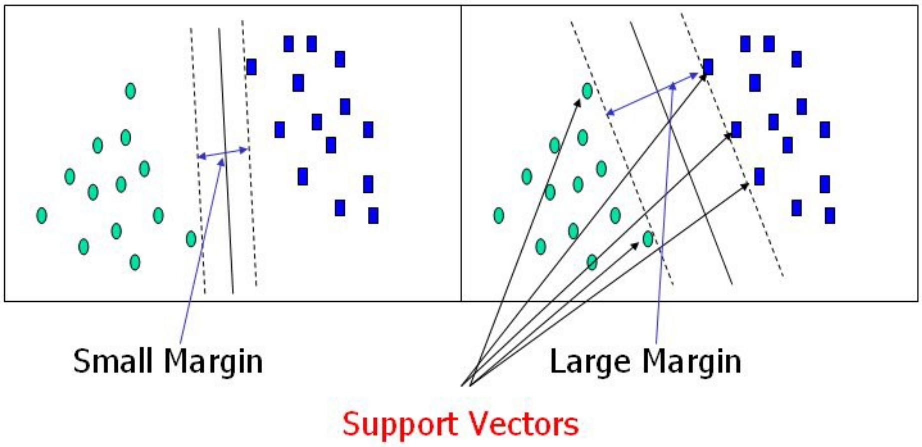


Figure 3.5: Support vectors

One more important SVM feature is the so-called "Kernel trick" [45].
All the previous examples show linearly separable data. The idea behind the kernel trick is that data, which isn't separable in our N-dimensional space may be separable in a higher dimensional space. One good example is shown in Fig. 3.6 where on the left we have two linearly nonseparable datasets, but on the right, when looked in another dimension we see that these classes are separable.

More details on how we trained our SVM classifiers will be presented in later Chapters.

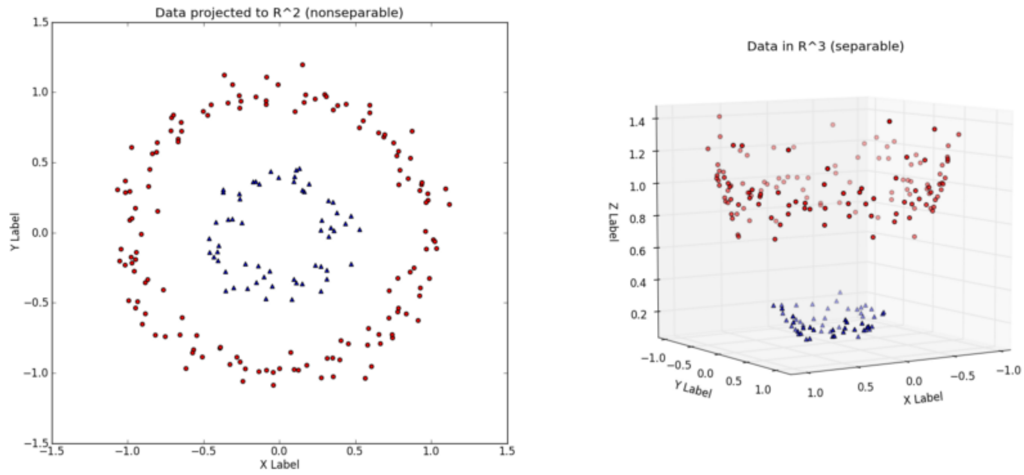


Figure 3.6: Linearly nonseparable data

Machine Learning, Neural Networks, and Deep Learning

Today, machine learning technology is a key component in many aspects of modern society. Machine learning systems are used for filtering web content, recommending content based on the user's interests, selecting relevant search results, processing images in our smartphones, and many more.

A machine learning algorithm is an algorithm that is able to learn from data. In [46] machine learning is defined with the following:

> *A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.*

Some tasks T often solved by using machine learning are:

- Classification - the computer program is asked to specify in which category some input belongs to
- Regression - the computer program is asked to predict a numerical value given some input
- Machine translation - the computer program converts/translates a sequence of symbols from one language to another
- Anomaly detection - In this type of task, the computer program sifts through a set of events or objects and flags some of them as being unusual or atypical
- Denoising - the machine learning algorithm is given as input a corrupted example $\tilde{x}$ obtained by an unknown corruption process from a clean example x. The learner must predict the clean example x from its corrupted version $\tilde{x}$.

A performance measure P is used to evaluate the abilities of a machine learning algorithm. The performance measure P is a quantitative measure of the performance of the algorithm and is often specific to the task T. For example, for a classification task the performance measure is often the accuracy or the error rate of the model calculated using the original data provided for training and the model's output. These kinds of design choices depend on the application [47].

Machine learning algorithms can be broadly categorized as *unsupervised* or *supervised* by what kind of experience E they are allowed to have during the learning process. In a supervised scenario, a model for machine learning is build based on a given dataset and the associated labels of the data. Depending on the task T these labels can represent different things. For a classification task, the labels will contain information about which data sample belongs to which class the model should learn to discriminate. In an unsupervised scenario, the labels are not present. In this case, the model tries to learn the underlying probability distribution of the data. Clustering is a good example of an unsupervised algorithm which divides the dataset into clusters of similar examples. Some machine learning algorithms do not just experience a fixed dataset. For example, *reinforcement* learning algorithms interact with an environment, so there is a feedback loop between the learning system and its experiences [47].

Using conventional machine learning limited the users because it required careful engineering and domain expertise to design a feature extractor that will transform the raw data into a suitable representation on which the learning process is based.

Deep Learning (DL) methods, on the other hand, are representation learning methods that allow a machine to be fed with raw data and to automatically discover the representations needed for detection or classification [48].

DL methods are, in their essence, neural network architectures, which is why deep learning models are often referred to as Deep Neural Networks (DNN). The term "deep" usually refers to the number of hidden layers in the neural network. Traditional neural networks consist of neurons that are organized in layers. Artificial neurons are the building blocks of neural networks and they are mathematical models which are designed to mimic their biological counterparts. A neuron produces a single output from its inputs x_i and weights ω_i associated with those inputs as shown in Fig 3.7

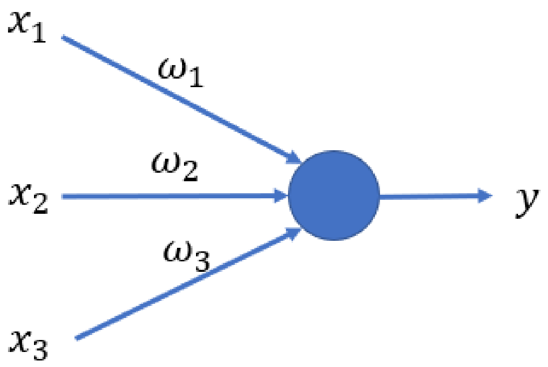


Figure 3.7: Mathematical model of an artificial neuron

Weights are real numbers expressing the importance of a given input to the output. The final output is defined as:

$$y = f(\sum_{i=1}^{n} x_i \omega_i) \tag{3.1}$$

where f is the neurons activation function. The activation function maps the resulting values into the desired range (e.g., from 0 to 1, or from -1 to 1). A sigmoidal activation function is most commonly used, but there are others. This is one more design choice that depends on the application. The neuron's output, 0 or 1, is determined by whether the weighted sum $\sum_{i=1}^{n} x_i \omega_i$ is less than or greater than some threshold value:

$$output = \left\{ \begin{matrix} 0 & if \sum_{i=1}^{n} x_i \omega_i \leqslant threshold \\ \\ 1 & if \sum_{i=1}^{n} x_i \omega_i > threshold \end{matrix} \right\} \tag{3.2}$$

One could say that a neuron weighs evidence given as input to make a decision given as the output. A complex network of neurons would have even more evidence which could be used to make complex decisions.

The former equation is further simplified by moving the threshold value to the left side of the inequality which leads further to notational simplifications when incorporating this into an optimization algorithm [49]. That being said, the former equation can be rewritten like:

$$output = \left\{ \begin{matrix} 0 & if \sum_{i=1}^{n} x_i \omega_i + b \leqslant 0 \\ \\ 1 & if \sum_{i=1}^{n} x_i \omega_i + b > 0 \end{matrix} \right\} \tag{3.3}$$

The parameter b is also called the *bias*, $b \equiv -threshold$. The bias is a measure of how easy it is to get the perceptron to output a 1. Or to put it in more biological terms, a measure of how easy it is to get the neuron to *fire* [49].

Fig. 3.8 shows a simple neural netowrk consisting of four layers.

On the left are the input neurons x_i making the input layer, and the rightmost layer is called the output layer which contains the output neurons. The two layers in between are called hidden layers. Conventional neural networks contain 2-3 hidden layers, while deep networks can have as many as 150 depending on the type of deep network and its applications.

When training a neural network in a supervised setting, our goal is to find the parameters (weights and biases) of the network such that the difference between the predicted labels $\tilde{y}$ and the true labels y is as small as possible. A function defining this difference, i.e. the error of prediction, is called the loss function which is a function of $\tilde{y}$ and y.

The loss function is used to quantify how well the training process is going. An optimization algorithm is used to find it's minimum. Often, the algorithm of choice is the Gradient

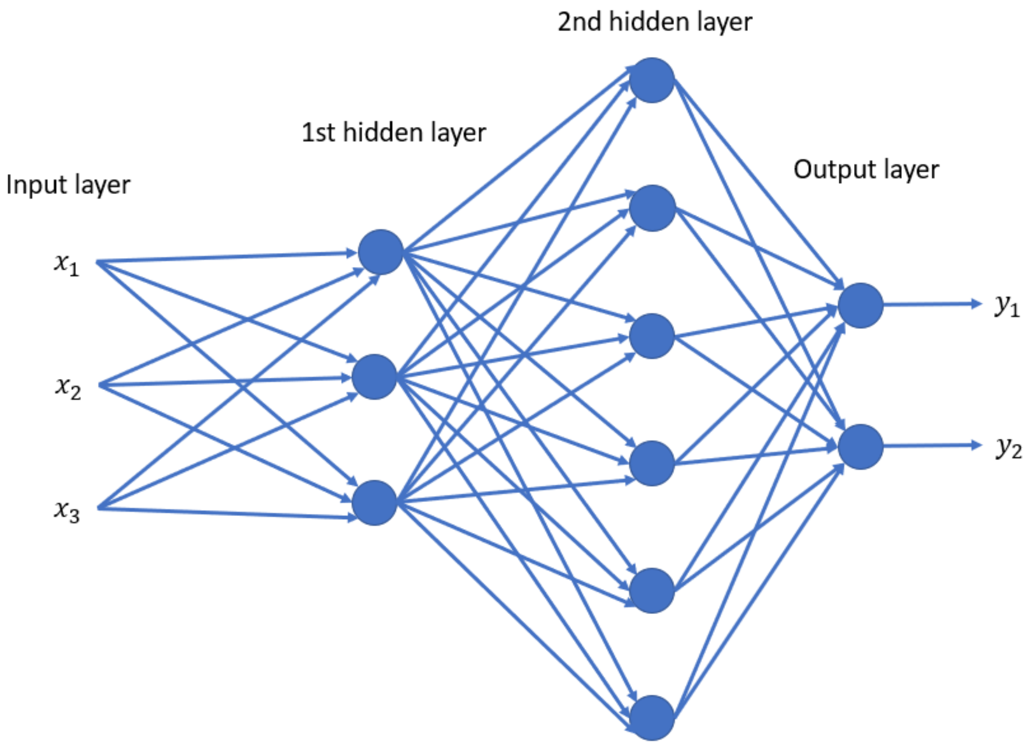


Figure 3.8: Artificial neural network with four layers

Descent [47, 48].

The learning process is an iterative one. The network's parameters are randomly initialized at the start of training. The output from one layer is used as input to the next layer in the network. Such networks are also referred to as *feedforward* neural networks. After the first pass, a prediction $\tilde{y}$ is calculated which is compared to the expected output y. The difference between the prediction and the ground truth is propagated back through the network in order to use that information to properly adjust the network's weights and biases. The process repeats itself until the network has achieved a satisfactory performance, i.e. until the parameters are found that minimize the loss function.

Gradient descent and backpropagation were also used to train deep networks. Unfortunately, except for a few special architectures, they didn't have much luck.

DNNs represent complex models which result in multidimensional and complex loss function that needs to be optimized. These networks would learn, but the process is slower, and in practice often too slow to be useful. Another issue with such complex loss functions is the presence of multiple local minima where the gradient descent algorithm could get "stuck".

Until recently, for that reason, it was believed too difficult to train deep multi-layer neural networks. Empirically, deep networks were generally found to be not better, and often worse, than neural networks with one or two hidden layers [50]. An approach that has been explored with some success in the past is based on constructively adding layers.

In 2006. a greedy layer-wise unsupervised learning algorithm for Deep Belief Networks

(DBN), a generative model with many layers of hidden causal variables was introduced that presented the breakthrough for training deep networks [51]. Upper layers are supposed to represent more abstract concepts that explain the input observation x, where the lower layers extract low-level features from x. They learn simpler concepts first and build on them to learn more abstract ones. In[52] three aspects of this strategy are considered important:

- pre-training one layer at a time in a greedy way
- using unsupervised learning at each layer in order to preserve information from the input
- fine-tuning the whole network with respect to the ultimate criterion of interest.

These ideas and concepts made it possible for successful training of deeper networks and for developing new techniques that even further optimized the whole process. These deep learning techniques are based on stochastic gradient descent and backpropagation, but also introduce new ideas. These techniques have enabled much deeper (and larger) networks to be trained and it turns out that these perform far better on many problems when compared to shallow neural networks. The reason, of course, is the ability of deep networks to build up a complex hierarchy of concepts.

DL methods aim at learning feature hierarchies where higher level features are formed by the composition of lower level features. Automatically learning features at multiple levels of abstraction allow a system to learn complex functions mapping the input to the output directly from data, without depending completely on human-crafted features. This is especially important for higher-level abstractions, which humans often do not know how to specify explicitly.

One of the most popular types of DNNs is known as Convolutional Neural Networks (CNN or ConvNet). A CNN convolves learned features with input data, and uses 2D convolutional layers, making this architecture well suited to processing 2D data, such as images.

We will go in more detail with CNNs in a later Chapter.

3.2.2 Postprocessing

Since detecting boundaries of vegetation areas is important for the intended application we perform per-pixel classification, i.e. classification is done for every pixel in an image. The resulting image containing two classes often contain pixels marked as vegetation in the non-vegetation class and non-vegetation pixels in the vegetation class. It is safe to assume that pixels or groups of pixels surrounded by vegetation also belong to vegetation. Also, solitary pixels or small groups of pixels surrounded by non-vegetation is often non-vegetation but is misclassified because it is too similar to underexposed vegetation (shadows under cars or parts of asphalt) or it has yellow reflectance. These gaps and solitary pixels can be removed after classification using a combination of morphological operations on the image.

Postprocessing is done in three steps:

- Morphological opening is applied using a circular structuring element with an experimentally determined radius.

  The morphological open operation is an erosion followed by a dilation, using the same structuring element for both operations [53].

  At this step, we presume that vegetation parts in the image are well connected and that those parts are large.
- Remove solitary groups of pixels of a certain size for which we presume are misclassified non-vegetation pixels.
- Fill up patches of a certain size for which we presume are misclassified as non-vegetation (surrounded by vegetation).

In order to perform the postprocessing, there are three parameters that we need to determine (radius of the structuring element, group size for removal and patch size for filling up). We experimentally select the ones with the best ratio of per-pixel accuracy and false detections.

3.2.3 Validation

To properly evaluate how a feature set will generalize to an independent data set we used 10-fold cross-validation to measure the performance and to validate the results obtained with our methods [54].

Generally speaking, in a *k*-fold cross-validation, the dataset is randomly divided into *k* equal sized subsamples. One subsample is used as the validation one for testing the model, and the remaining $k-1$ subsamples are used for training. The cross-validation process is repeated *k* times, i.e. until every subsample was used as the validation one. The *k* results can be averaged to produce a single estimation. The advantage of this method is that all observations are used for both training and validation, and each observation is used for validation exactly once.

The purpose of testing this way is to exclude the possibility of choosing “the perfect” training samples that would not generalize well to new data. In every run, the used samples are selected randomly, but paying attention to special case samples.

3.3 Classification methods

The first, simplest approach we can take is to just use color (vegetation is "mostly" green, right?) as a discriminator. We expected that this approach will not yield promising results but it did indicate some potential problems in choosing the training data which was then corrected accordingly. We will cover this in greater detail in the following chapters.

Once we concluded that using only color is not going to be good enough, a new feature, describing the texture, needed to be added. This presented a new set of obstacles. Texture features are one of the most investigated areas in image processing, so we started with a simpler approach and build up to a more complex texture descriptor.

First, entropy was used to describe the texture. Entropy [55] is a measure of chaos in some sense, and vegetation areas, when compared to the road or other roadside objects contain more chaos.

Adding texture improved results compared to using only color (as expected), but additional steps were used as postprocessing to further improve results.

The next step was to use a more complex texture feature that would be invariant to rotation and scale. Any patch of vegetation should be recognized as vegetation region regardless of the orientation. Given the limited dataset, rotation invariance is a desirable feature for any method. Also, the appearance of the vegetation, i.e. the scale, depends on the distance from the camera and our goal is to detect all kinds of roadside vegetation from thin grass to wider leaves of bushes regardless of the distance from the camera.

There are many different applications of vegetation detection techniques and they all vary in types of features used and classifiers trained. In order to achieve good results with conventional machine learning techniques, the right features have to be selected.

As the final step in our research, a Convolutional Neural Network was implemented and tested. With CNNs the features are learned from the input data so we do not have the problem of selecting the feature set as we did before.

3.3.1 Classification based on color features only

Human perception of color is one of the most important visual elements which helps us recognize different objects. Transferring this human ability to a computer algorithm is far from simple. One major problem is the change in intensity and color in different lighting conditions which does not pose a problem for the human visual system, but for a computer algorithm, a change in luminance can result in large differences in calculated features.

Color spaces

As it was introduced in Chapter 2.1, some authors use different color spaces as color features for vegetation detection (e.g. HSV [11], YUV [17, 18]) in order to improve results, while others, ignore the luminance component completely in order for their feature set to be less sensitive to lighting changes in the scene. One such example is presented in [11] where the authors chose to use only H and S components, ignoring the luminance component V from the HSV color space. We took that into consideration and evaluated different color spaces to determine if a different color representation performs better for the problem at hand.

We used four different color spaces for testing: RGB, HSV, YUV, and CieLAB. CieLAB was added because it is designed to mimic human perception of color. Additionally, we tested only the H and S components excluding the lightness from HSV and only A and B components excluding the L component from CieLAB.

Vegetation indices

The NDVI vegetation index was mentioned several times in Chapter 2 since it is one of the most used features for vegetation detection in satellite images. This index and its derivatives can only be calculated if the needed image modalities are available. For our case, when using only the visible part of the spectrum, we turn to the two available vegetation indices based only on R, G and B values.

The Visible Vegetation Index (VVI) and the Green-Red Vegetation Index (GRVI) are indices that use only information from the visible spectrum, combining color channels to obtain a single feature. For example, VVI is a measure of the amount of greenness of an image [19]:

$$VVI = \left[\left(1 - \left|\frac{R-R_0}{R+R_0}\right|\right)\left(1 - \left|\frac{G-G_0}{G+G_0}\right|\right)\left(1 - \left|\frac{B-B_0}{B+B_0}\right|\right)\right]^{1/w} \tag{3.4}$$

while in [20] authors evaluate the use of this index as a phenological indicator.

$$GRVI = \frac{G-R}{G+R} \tag{3.5}$$

where R, G, and B are reflectances of visible red, green and blue respectively. $[RGB]_0$ is a reference green color vector and w is a weight exponent used for adjusting the sensitivity of the scale. The final classification into vegetation and non-vegetation is performed by thresholding. The threshold for vegetation detection from the calculated GRVI is zero, while the threshold for VVI is a parameter that is experimentally determined.

Results

Nine different feature vectors containing different color features were used to train an SVM classifier with a radial kernel. This classifier was used only in this stage of testing to determine if there is a significant difference between these features and to serve as an aid in selecting the best one.

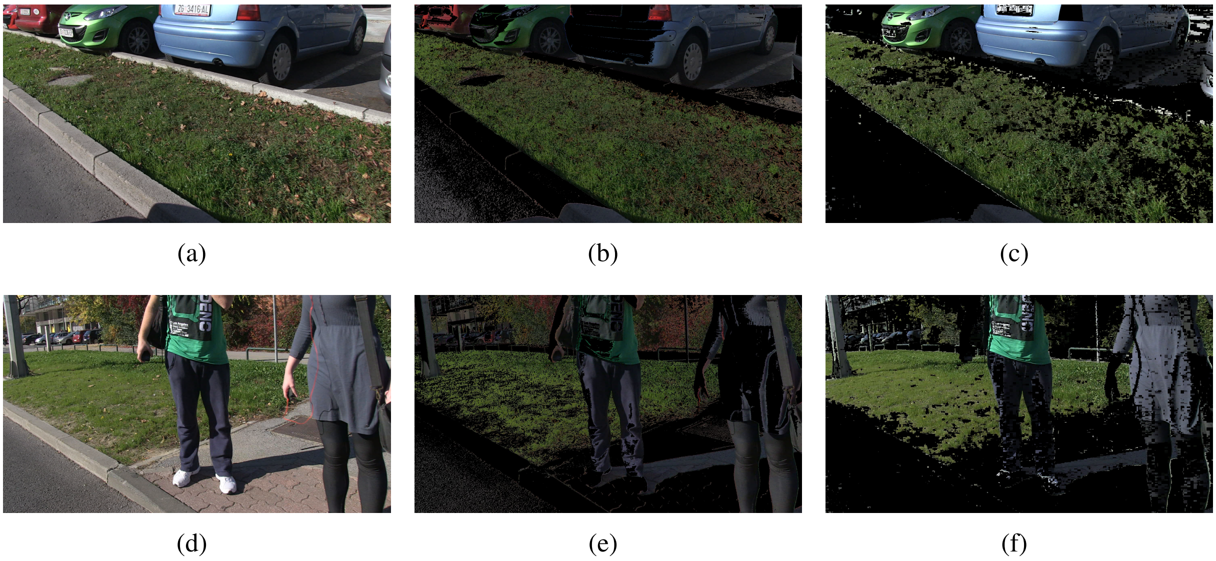
The results of running 10-fold cross-validation are shown in Table 3.1. From these results, we conclude that using or excluding the lightness component from the feature set does not affect the results drastically. Slightly better performance can be seen when using the lightness component with color information, i.e., better accuracy is achieved using HSV and CieLAB then HS and AB features respectively. The second conclusion we draw from Table 3.1 is that CieLAB, YUV, and HSV color spaces perform similarly and are giving slightly higher accuracy results compared to RGB which was expected due to the additive nature of RGB and the fact that every component contains luminance information which makes it more sensitive to luminance changes.

Table 3.1: Comparison of different color spaces used for vegetation detection.

Feature vector	**Accuracy**	**Feature vector**	**Accuracy**
CieLAB	91,9015%	AB	89,9947%
YUV	91,8235%	UV	61,7462%
HSV	91,4285%	HS	88,7151%
RGB	89,6314%		
VVI	58,342%		
GRVI	67,617%		

Finally, thresholding of the vegetation indices VVI (Fig. 3.9(b) and Fig. 3.9(e)) and GRVI (Fig. 3.9(c) and Fig. 3.9(f)) did not yield better results compared to the "simpler" color features. This is not surprising due to lack of information in these vegetation indices. They use only color information which is not enough to properly describe and to afterward classify objects similar in color to vegetation. Problematic objects in images like the green automobile in Fig. 3.9(a) and the green T-shirt in Fig. 3.9(d) are partially(Fig. 3.9(b) and Fig. 3.9(e)) or completely misclassified(Fig. 3.9(c) and Fig. 3.9(f)) when using the VVI and GRVI-based methods. Even parts of the surrounding objects which are not similar to vegetation in color are also misclassified as vegetation.

The threshold for vegetation detection from the calculated GRVI is zero, while the threshold for VVI is a parameter that is experimentally found. Calculating VVI requires adjusting four more parameters: $[RGB]_0$ and w which does not make it robust. The same set of parameters



(a) (b) (c)

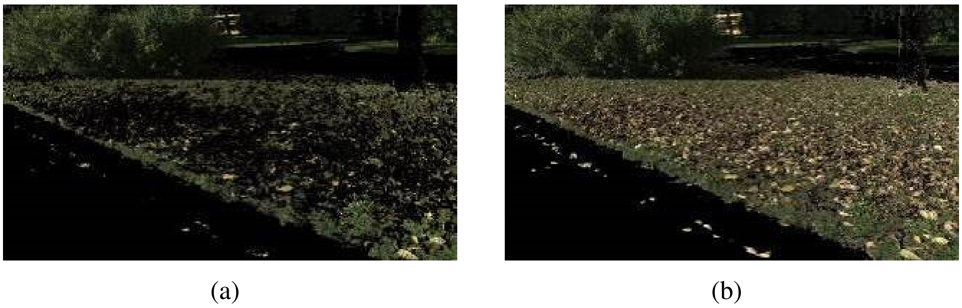
(d) (e) (f)

Figure 3.9: Result example of methods based on VVI and GRVI for some problematic images. The results are achieved by thresholding the vegetation indices: (a) Original image, (b) VVI, (c) GRVI, (d) Original image, (e) VVI, (f) GRVI

cannot be used for scenes that differ in the amount of light or shade they are getting. Our results show that these indices need additional modification for usage in ground recordings.

Problems

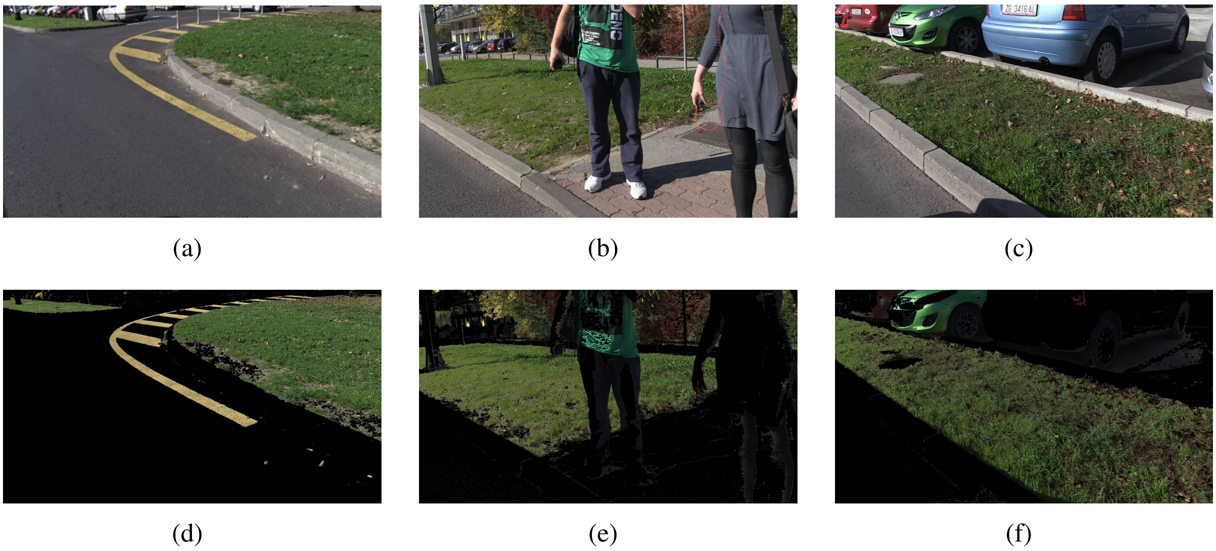
As mentioned in Chapter 3.1.1, using only green vegetation in training will make a classifier that does not detect dry vegetation, i.e. vegetation that is not green like in Fig. 3.10(a). These results improve when we add these samples in the training process as it is shown in Fig. 3.10(b).


(a)

(b)

Figure 3.10: Detection using: (a) only green vegetation for training, (b) all vegetation for training.

While including dry vegetation in the training phase will make a more universal classifier, on the other hand, the number of false positive detections will increase. Beside the green objects like the green car in Fig. 3.11(f) and the green T-shirt in Fig. 3.11(e) that are falsely detected as vegetation, the yellow markings on the road in Fig. 3.11(d) are also falsely classified as vegetation by a color-based classifier.


(a)

(b)

(c)

(d)

(e)

(f)

Figure 3.11: Examples of vegetation detection using only green vegetation for training.

This is a good demonstration of how classification results are affected by the selected training set. We expanded our classifier to detect dry vegetation, i.e. roadside vegetation of different colors and/or states, but at the same time, we have increased the number of false positives. In order to solve this problem, we need to add a new feature to our feature vector since using only color is evidently not enough.

3.3.2 Color and texture features

Features

As shown in the previous chapter, using only color as a feature had some expected drawbacks so objects similar to vegetation in color were classified as vegetation. To solve this problem we decided to add texture features to the feature vector used for training the classifier. There are many different methods for calculating texture features in images and choosing one is a difficult task. Knowing that vegetation is diverse and that vegetation parts in images contain more information then homogeneous surfaces we decided to use entropy as a texture feature. Entropy is a statistical measure of randomness and it can be described as a measure of the amount of disorder in a system. For images, it can be expressed as a spread of states (gray levels) which the individual pixel can adopt. If pixels in an image, or in a part of an image, have the same values then the entropy is zero. On the other hand, if an image (or a part) contains pixels with varying values, the entropy will be higher. We expect that vegetation regions in an image have high entropy. The entropy H of an image is defined as [55]:

$$H = -\sum_{k=0}^{M-1} p_k log_2 p_k \tag{3.6}$$

where M is the number of gray levels and p_k is the probability associated with a gray level k.

Feature ranking

Usually, entropy is calculated using a grayscale image, but since grayscale images are sensitive to changes in luminance we also considered calculating entropy in different color spaces. Color spaces taken into consideration were the same ones used for testing color features. We used feature ranking to select in which modality to calculate the entropy.

We created a feature vector that contains color features (LAB) and 9 texture features that are entropy calculated for R, G, B, H, S, V, L, A and B components of the corresponding color space models. Twelve features is not a big number for any kind of feature selection, but feature ranking is a useful filtering method and a good preprocessing step [42]. All these features are ranked and based on the calculated rank we choose the best one as our descriptor.

This test was repeated 10 times in order to establish statistical significance. In every iteration, the color features were always the top three ranked features. Entropy calculated for saturation is the only calculated entropy that was mostly top-ranked while others varied drastically from test to test. That is why only entropy of this channel was added to the feature vector.

Training the classifier

SVM was used as the classifer. For training and classification, besides the linear kernel, the quadratic and radial kernels were tested in order to take advantage of SVMs kernel trick and to transform features in a new feature space. The best classification accuracy was obtained using the radial basis SVM.

The data is normalized before training, and training is done using the Gaussian Radial Basis Function kernel with the scaling factor (Sigma) set to 1. Sequential Minimal Optimization is the method used for finding the separating hyperplane.

Postprocessing

The classification is done on the pixel level which leaves room for morphological operation-based postprocessing as described in Chapter 3.2.2.

For performing the morphological opening a circular structuring element with a 5-pixel radius was used. We remove groups that are less than 3000 pixels and the filling was done for patches less than 500 pixels big. The small value of 500 was used because there are parts of roadside vegetation that have utility shafts that are small and any bigger size of 500 would misclassify these parts as vegetation (e.g. utility shafts in Fig. 3.12(d) and Fig. 3.13(d)).

Experiments and results

To find the optimal parameters for this method we tested several options for every aspect and choose the ones that gave the best per-pixel accuracy.

Entropy is calculated for every pixel based on its neighborhood. We tested four neighborhood sizes: 7x7, 9x9, 11x11, and 13x13. Experiments showed that the optimal neighborhood size for this application is 9x9. Taking smaller block sizes resulted in lower per-pixel accuracy, and using a bigger neighborhood did not improve the final result.

Adding texture features improved detection in these problematic images. With these features the green objects in Fig. 3.12(a) and Fig. 3.12(b) and the yellow stripes in Fig. 3.12(c) are much better classified as non-vegetation (Fig. 3.12(d) - 3.12(f)) compared to Fig. 3.11 where only color features were used.

Figures 3.12(d) - 3.12(f) show the classifier output. In these images the misclassified solitary pixels and groups of pixels can be seen which are removed in the postprocessing step (Fig. 3.12(g) - 3.12(i)).

The trained classifier achieved an average accuracy of pixel classification of **92,436**% prior to performing the postprocessing step. The morphological operations improve the overall accuracy to **94,995**%.

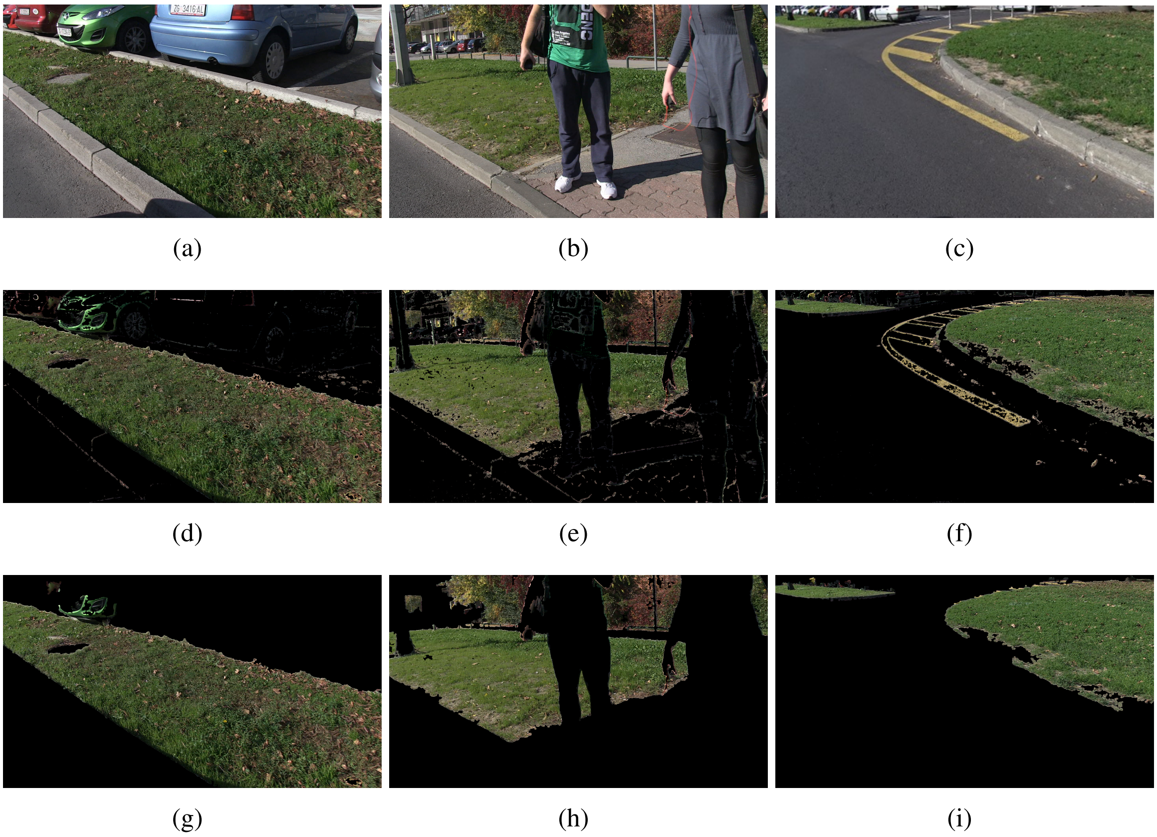


Figure 3.12: Classification result of problematic areas: (a)(b)(c) Original image, (d)(e)(f) Classification result of SVM - before postprocessing, (g)(h)(i) Final image result - after postprocessing

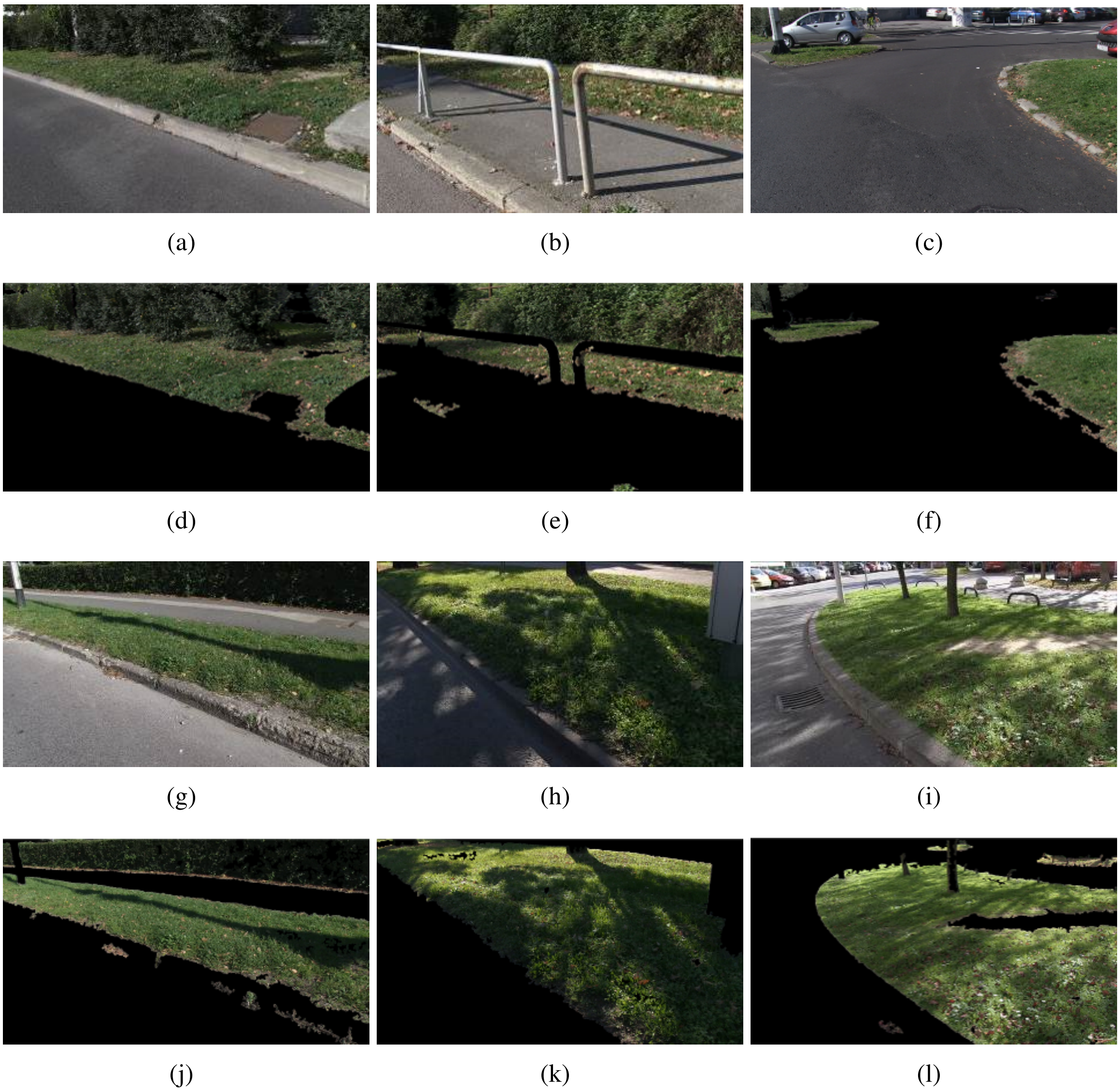
Additional vegetation detection results using the method presented in this chapter are shown in Fig. 3.13.

Fig. 3.13(d), Fig. 3.13(e) and Fig. 3.13(f) show good detection results in different traffic scenes, while good performance in detecting vegetation in shade can be seen in Fig. 3.13(j), Fig. 3.13(k) and Fig. 3.13(l).

Problems

Even though the overall per-pixel accuracy was increased by adding a texture feature in the training process there were still some issues present so further improvements were needed. The main issue with this method is:

- In Fig. 3.12(d) and Fig. 3.12(g) parts of the green car (mostly edges and places of high reflection on the hull) are still detected as vegetation even though we included these examples in the training set. This color is too similar to the vegetation examples in the training set and because entropy is high over the edges the classifier decided that these parts are also vegetation. In order to improve the accuracy of the classifier, we will consider using a more complex feature for describing vegetation texture. Using only entropy is not enough due to the nature of the images we are processing. Real life traffic scenarios



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

(j) (k) (l)

Figure 3.13: Examples of detection in various traffic environments: (a)(b)(c)(g)(h)(i) The original image; (d)(e)(f)(j)(k)(l) The result of classification

will often contain similar objects in color to vegetation that could have high entropy due to uneven texture and/or surface.

3.3.3 Color and CWT-based texture features

So far we have observed that we need both color and texture features in order to train a good classifier. We also concluded that we need a more descriptive texture feature in order to properly classify non-vegetation areas which are similar to vegetation in color and entropy-based texture feature.

The Continuous Wavelet Transformation [56] is a filtering technique that can be interpreted as a measure of the similarity between a signal and an analyzing function. In our case, the signal is a 2D image and the analyzing function is a wavelet, and selection of a proper wavelet function depends on the application. There are many wavelet families and each one has certain advantages and disadvantages. In short, the 2D CWT compares the image to the shifted and compressed/stretched versions of the selected wavelet which makes this technique robust to scale and orientation [56]. Since we need to detect vegetation of various types and various sizes this robustness might improve the classification results.

Additionally, we tested this approach with a preprocessing step specific for the intended usage of a vegetation detection algorithm for roadside maintenance. We assume that the recording of roadside vegetation will be continuous from a moving vehicle, which means that we could focus our detection on the area closer to the vehicle. Vegetation that is further away from the vehicle will be detected later as the vehicle approaches it. The preprocessing step of our method is meant to help determine a region of interest for vegetation detection. This way, we narrow down the classification area in each frame of the recorded video.

More about the 2D CWT as a texture feature and how we selected the area of interest (AOI) will be described in the following chapters.

With the new, preprocessing step, the method presented in this chapter consists of four steps as it is shown via block diagram in Fig. 3.14:

1. preprocessing,
2. extraction of selected features,
3. pixel classification and
4. postprocessing.

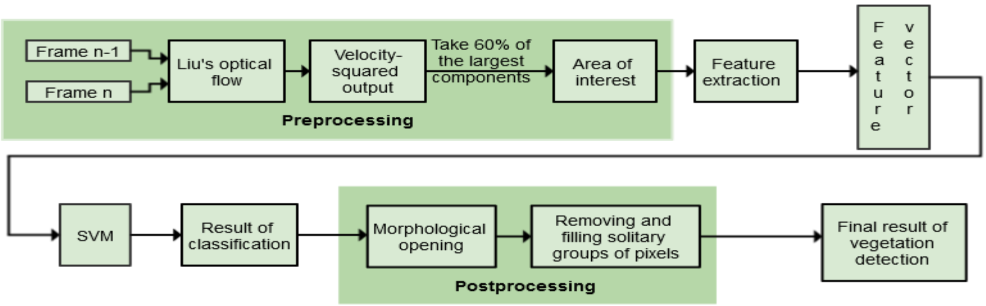


Figure 3.14: Block diagram of the proposed method

Preprocessing

The area of interest is determined for each frame and then vegetation is detected in that area. This is necessary because objects that are further from the camera lack detail and sharpness compared to objects that are closer. Our method for vegetation detection depends on texture features which can be different for the same object, depending on the distance from the camera.

To determine a region of interest a simple window of a certain size could be applied but that approach is not universal and it fails in certain traffic scenarios. Highways and main roads are usually surrounded only by grass, but through the city, the roadside landscape changes. Somewhere there is only grass by the road, but somewhere there are trees and bushes of different sizes as shown in Fig. 3.15. To cover these situations we needed to consider a more universal and complex method for determining the distance from the vehicle.

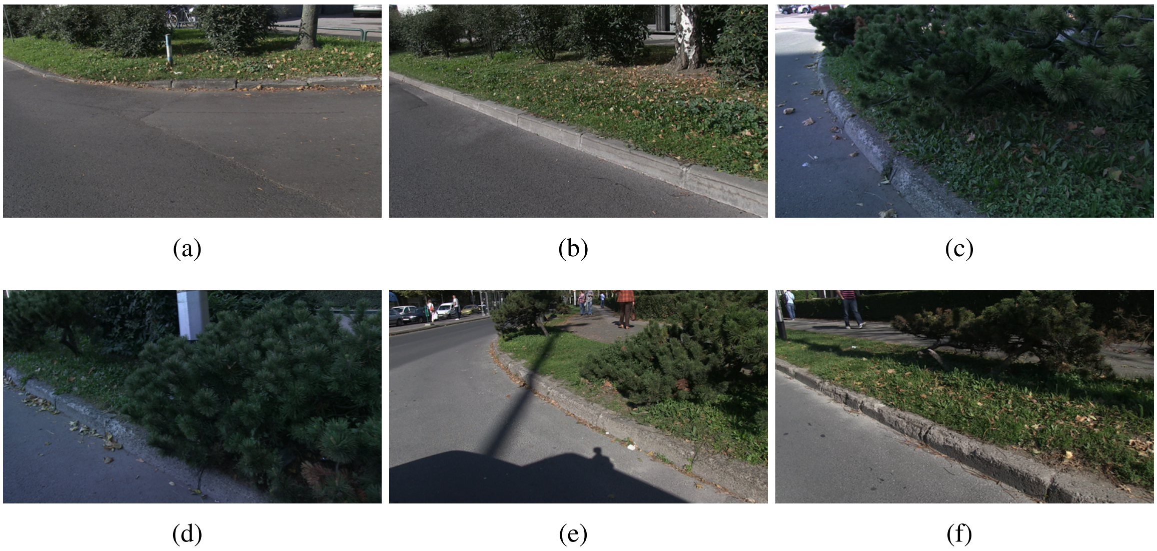


Figure 3.15: Examples of different roadside vegetation to consider when determining the area of interest

Estimating depth, i.e. distance, from images is usually done with a pair of calibrated stereo cameras. Monocular depth estimation either considers a supervised learning approach [57] or a specific camera set up (static and/or calibrated camera, the camera's optical axis coincide with the vehicle translation direction [58], etc.). Since we only have our database for development which is recorded with one camera and which does not meet the requirements of abovementioned methods, we turn to a simpler method of estimating distance as optical flow. Optical flow estimates the direction and speed of an object's motion from one frame to another. By estimating optical flow between video frames the velocities of objects in a video can be measured. In general, if the camera is moving, objects closer to the camera will display more apparent motion. By selecting the area in the scene where larger motions are present we determine our area of interest. Optical flow does not give us precise values of distances to objects but for our purpose, a relative relation of distances will suffice.

Traditional methods for computing optical flow like Horn-Schunck [59] or Lucas-Kanade [60] did not yield satisfying results for our data so we decided to use a more complex implementation developed by C. Liu in his doctoral thesis [61]. The algorithm is based on [62] and [63]. In [62] the authors propose a novel approach that integrates several concepts for computing optical flow. They combine three assumptions: a brightness constancy assumption, a gradient constancy assumption which makes the method robust against gray value changes, and a discontinuity-preserving spatiotemporal smoothness constraint. In [63] authors propose a combined local-global method that combines robustness to noise of local methods with global approaches that yield dense flow fields.

Feature Extraction

For this method, we focused more on finding a texture feature that would better fit our data.

There are many different methods for calculating texture features in images. Since we are interested in texture features that include orientation of the texture, we decided to use a two-dimensional Continuous Wavelet Transform (2D CWT) which is characterized by a rotation parameter, in addition to the usual translations and dilations [56].

The 2D CWT is a decomposition of an image on a set of dilated, rotated and translated versions of a single function called the mother wavelet. The scale dependence allows sensitivity to variations in resolutions, while the rotation dependence leads to robust behavior under varying target orientations. 2D CWT is a linear transformation and is easily understood from its definition as a convolution:

$$S(\overrightarrow{b}, a, \theta) = \int \psi^* \left(r_\theta \left(\frac{\overrightarrow{x} - \overrightarrow{b}}{a} \right) \right) s(\overrightarrow{x}) d^2x \tag{3.7}$$

where $s(\overrightarrow{x})$ is the input grayscale image and ψ is the analyzing mother wavelet, which is translated by $\overrightarrow{b}$, dilated by a and rotated by an angle θ (r_θ is the rotation operator) [56].

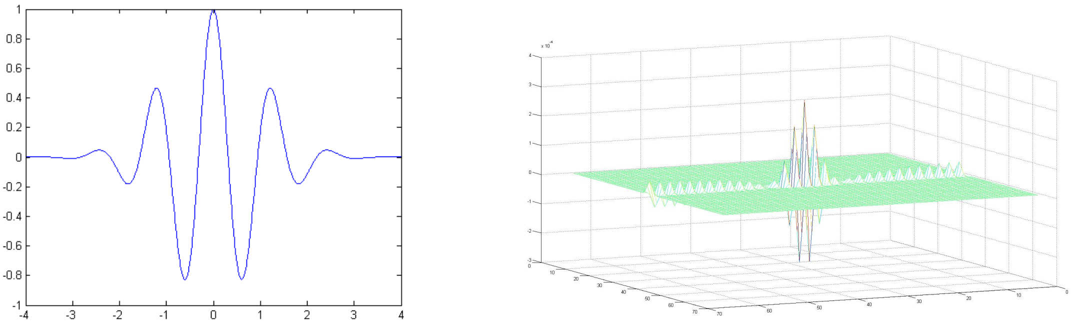
The type of wavelet chosen depends on the precise application. We considered using isotropic wavelets instead of anisotropic oriented wavelets but initial experiments showed better performance when using oriented wavelets which are required for detecting patterns in different directions. Several oriented wavelets were considered and tested but the oriented Morlet wavelet showed the best performance, i.e. the lowest error rate in falsely detecting green objects as vegetation. The Morlet mother wavelet in frequency domain is defined as:

$$\psi(\omega_x, \omega_y) = e^{-\sigma^2 \left((\omega_x - \omega_0)^2 + \frac{(\varepsilon \omega_y)^2}{2} \right)} \tag{3.8}$$

where σ is the spread factor, ω_0 is the shift in frequency domain and ε is the anisotropy factor. The effective support of the function $\psi(\omega_x, \omega_y)$ is an ellipse centered at ω_0 and elongated in the

ω_y direction, thus contained in a convex cone, that becomes narrower as ε increases [56].

The Morlet mother wavelet is shown in Fig 3.16.



(a)

(b)

Figure 3.16: Morlet mother wavelet: (a) in 1D, (b) in 2D

Using the Morlet oriented wavelet for 2D CWT gives us five free parameters for tuning: scales a, orientations θ, σ, ω_0 and ε. Modifying these parameters will dictate how the wavelet looks like, which will affect the types of features being detected. For example, if we modify the scale or the orientation parameters we will have better detection of these scales and rotations respectively. The changes in the anisotropy factor ε will result in varying magnitude as shown in Fig. 3.17.

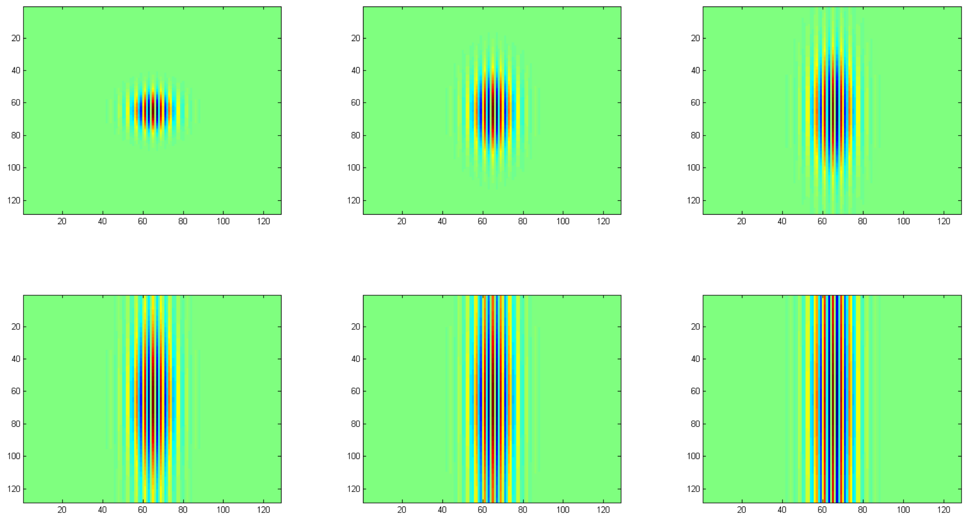


Figure 3.17: The effect of changing the ε parameter

Increasing σ will increase the number of waves as shown in Fig. 3.18.

Lastly, increasing ω_0 will increase the width of the mother wavelet as shown in Fig. 3.19.

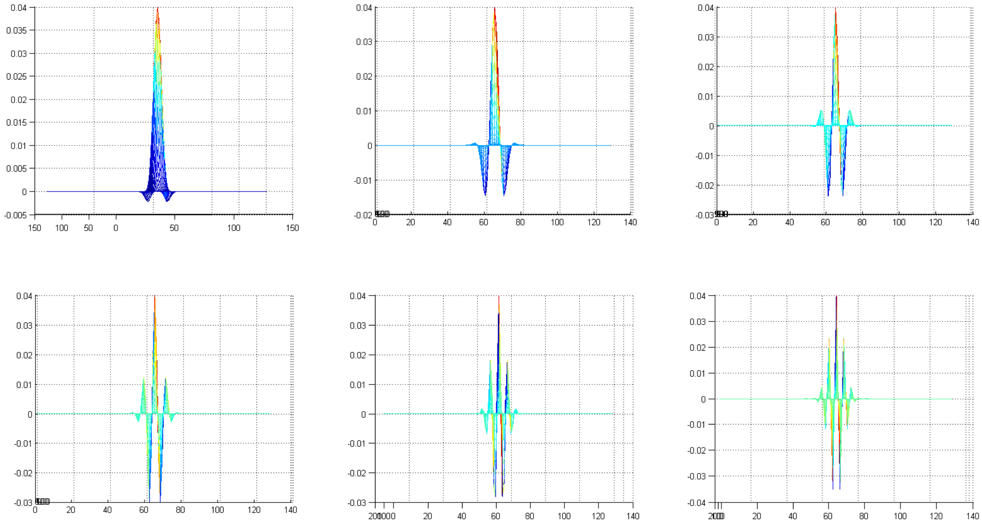


Figure 3.18: The effect of changing the σ parameter

Feature Ranking

The same method for feature ranking as described in 3.1.2 was also used here to extract the best ranking features. As we already mentioned, using the Morlet oriented wavelet for 2D CWT gives us five parameters for tuning. Increasing ω_0 and σ increases the number of waves and the width of the mother wavelet function respectively. More waves are required in order to detect image regions where the same pattern (e.g. leaves and grass) repeats several times. The anisotropy factor ε does not influence this property and was therefore set to 1. For selecting scales a, rotations θ, σ and ω_0 we used feature ranking. A feature vector containing the selected color features and wavelet coefficients with *scales* ranging from 1 to 10 with an arbitrary increment of 0.5 and *orientations* from 0 to $\frac{7\pi}{8}$ with an increment of $\frac{\pi}{8}$ was ranked.

For statistical significance, ranking was repeated 10 times on different sample sets. L, A and B features were always the three top ranking features which indicate the importance of color features. Small scales up to 5 and scales larger than 9 always had the smallest scores and that is why we decided to use the scale range from 5 to 9. Best ranked orientations were $orientations = [\frac{\pi}{8}, \frac{2\pi}{8}, \frac{6\pi}{8}, \frac{7\pi}{8}]$ which was as expected because orientations around 0 and $\frac{\pi}{2}$ are more common in non-vegetation segments in images than the other orientations.

After finding the best scales and rotations ranking was done for features with varying σ and ω_0. Best ranked were $\sigma = 2$ and $\omega_0 = 6$ which were used in our method.

Finally, we used the oriented Morlet wavelet with selected parameters obtained via ranking and filtered the original image for nine selected scales and for four selected rotation angles. We summed up the CWT coefficients for every scale and rotation and thus got one feature which holds information from all selected rotations and scales. The calculated texture feature was

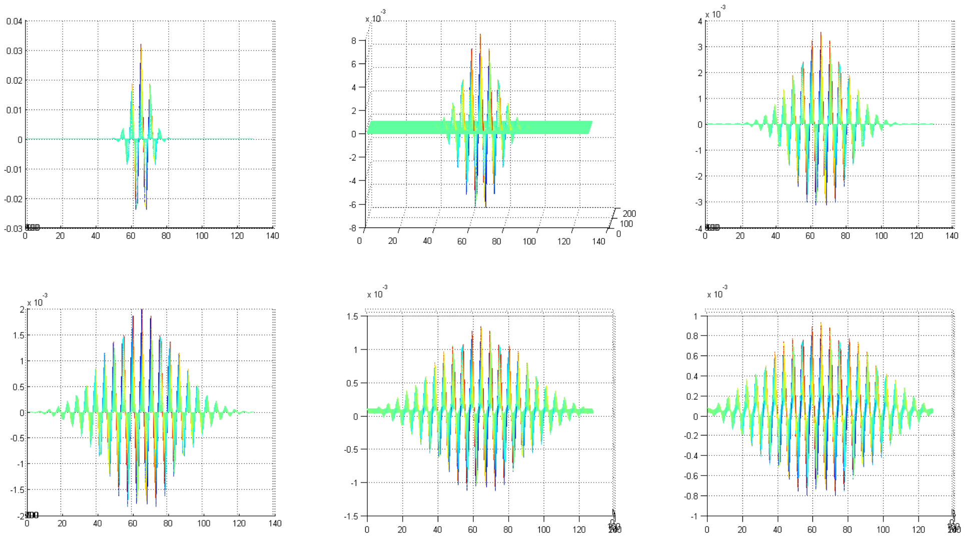


Figure 3.19: The effect of changing the ω_0 parameter

added to the feature vector alongside color features.

Training the Classifier

For the classifier, we, again, selected an SVM model with the radial basis kernel to train.

The data is normalized before training, and training is done using the Gaussian Radial Basis Function kernel with the scaling factor (Sigma) set to 1. Sequential Minimal Optimization is the method used for finding the separating hyperplane.

Postprocessing

The classification is done on the pixel level which, again, leaves room for a morphological operation-based postprocessing step similar to ones used before. Postprocessing is done in three steps as described in Chapter 3.2.2.

For achieving the final results we used a circular structuring element with a radius of 3 pixels, we remove groups that are less than 3000 pixels and filled patches less than 750 pixels.

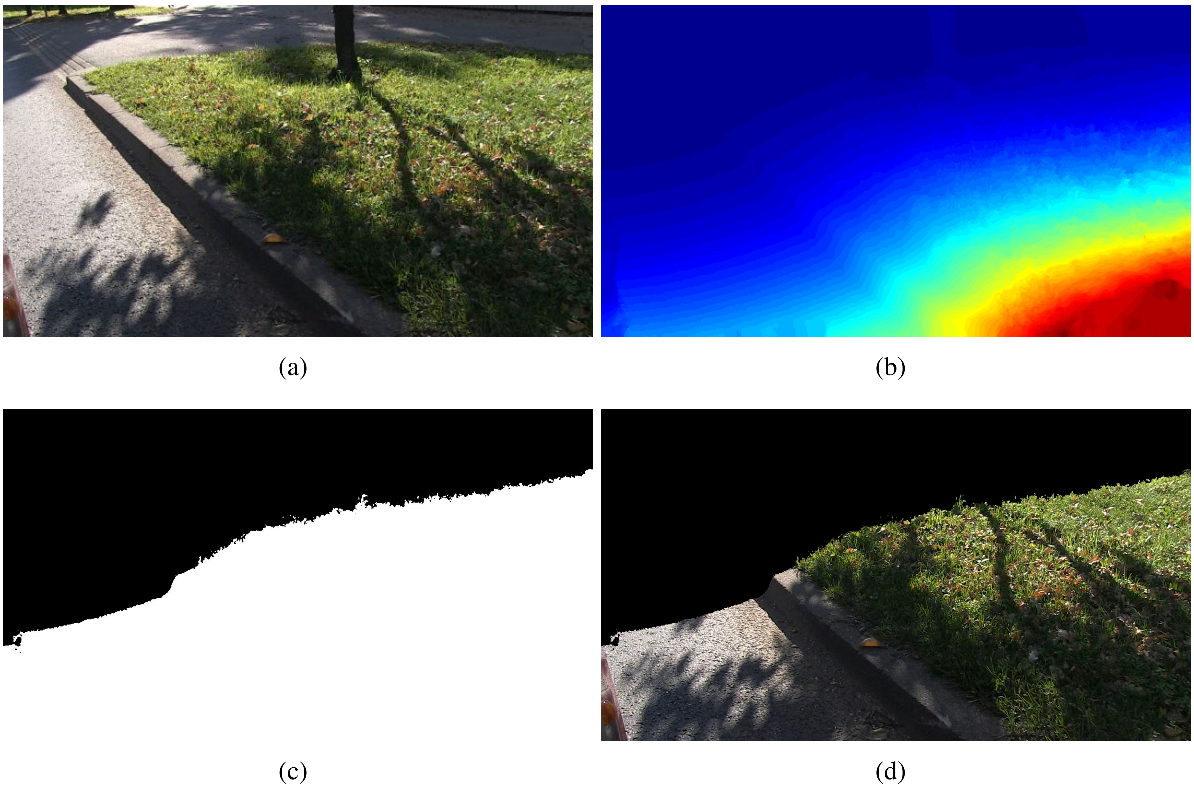
Experiments and Results

The optical flow computation returns the horizontal and vertical components of the flow field computed from two consecutive frames. We used these components to generate a velocity-squared output. From this we chose our area of interest based on the calculated amounts of motion. Because our database contains recordings with varying views (the camera's tilt changes) we had to find the optimal amount of motion that will be suitable for determining the area of interest in all these images. Our experiments have shown that 60% of the components with

Table 3.2: Classification Results.

Feature vector	**Accuracy**
AB	$89,9947\%$
CieLAB	$91,9015\%$
CieLAB + 2D CWT	$93,89\%$
CieLAB + 2D CWT + Pre-processing + Post-processing	$96,10182\%$

highest motion is the optimal amount for our problem. The described process of determining the area of interest is shown in Fig. 3.20.


(a)

(b)

(c)

(d)

Figure 3.20: Determining the area of interest: (a) Original image, (b) Velocity-squared output calculated from optical flow, (c) 60% components with highest motion, (d) Final area of interest

Adding the texture feature did not only improve detection in problematic images (Fig. 3.21(d)), but it improved the overall accuracy of per-pixel classification compared to using only color features as seen in Table 3.2.

With these features the green object in Fig. 3.21(a) is much better classified as non-vegetation seen in Fig. 3.21(d) compared to Fig. 3.21(c) where only color features were used. In Fig. 3.21(e) we show the vegetation class labeled in the area of interest in which the misclassified pixels can be seen. To correct this we are using the postprocessing step. The result of postprocessing is shown in Fig. 3.21(f).

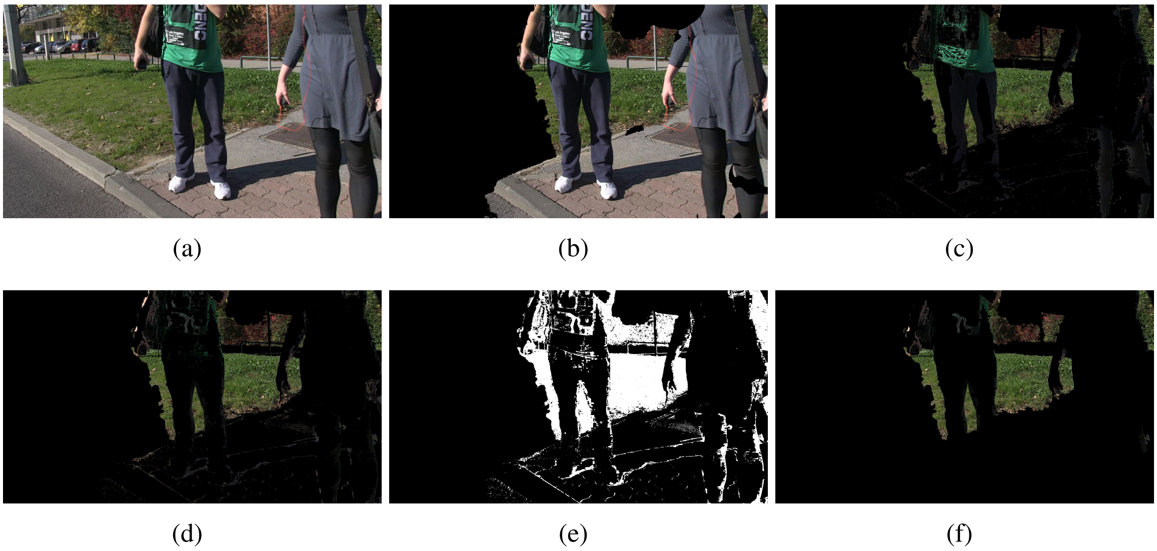
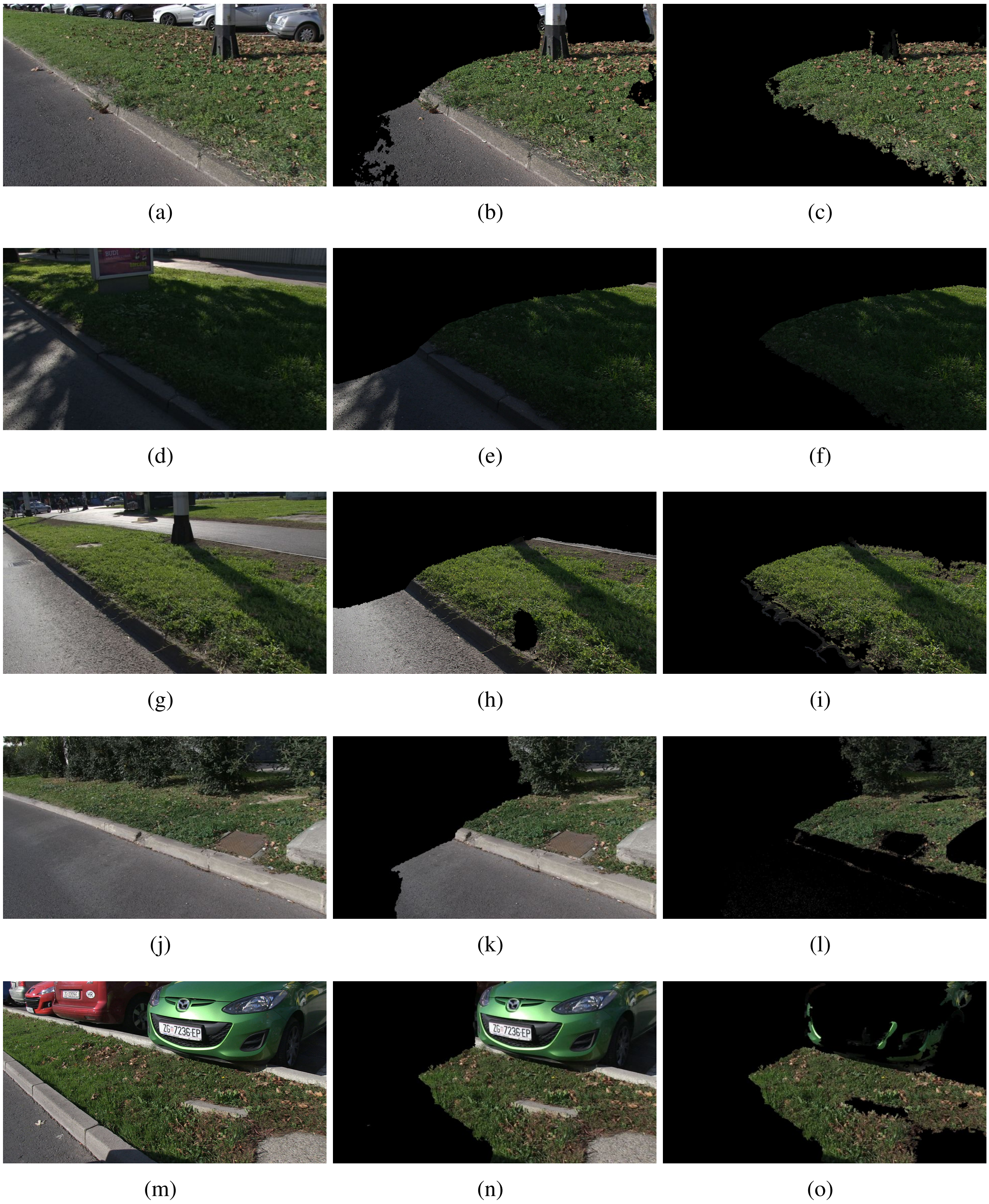


Figure 3.21: Detection example for a problematic image: (a) Original image, (b) Area of interest, (c) Detection using only color features, (d) Detection using color and texture features, (e) Vegetation class labeled, (f) Result after classification and postprocessing

Some additional results of vegetation detection using the method presented in this chapter are shown in Fig. 3.22. The advantage of using optical flow for determining the area of interest instead of using a fixed window can be seen in Fig. 3.22(k). Using the same window for Fig. 3.22(g) and Fig. 3.22(j) would not give correct areas of interest in both cases. A fixed window would not include the bushes in Fig. 3.22(j) in our area of interest, although it should. Good performance of detecting vegetation in shade can be seen in Fig. 3.22(e) and (h). One more example of correctly detecting green object as nonvegetation can be seen in Fig. 3.22(o). The edges of the car are falsely detected because edges have high information content and differ from the smooth areas of the car. The color features were the best ranked ones and that is why little difference in calculated texture features will lead to misclassification. Although the green car is not completely correctly classified we are satisfied with this result because without using any texture features the whole car is detected as vegetation. Using a preprocessing step did help in finding our detection area but it has its drawbacks. We see in Fig. 3.22(b), (h), (k) and (n) the optical flow did not perform perfectly. The small patches in Fig. 3.22(b) and (h) are easily fixed in the postprocessing step. In Fig. 3.22(k) and (n) the asphalt is labeled as not in our area of interest. This happens because the asphalt's lack of detail makes it difficult to match corresponding features between two frames, so no motion is detected. These mistakes are not critical to our application because the vegetation parts of the image have greater detail and those parts of the image are correctly labeled.



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

(j) (k) (l)

(m) (n) (o)

Figure 3.22: Additional results of vegetation detection: (a) The original image; (b) AOI with the correctly included lamppost; (c) Detected vegetation in the AOI; (d) The original image; (e) AOI with good performance in shade; (f) Detected vegetation in shade in the AOI; (g) The original image; (h) AOI with a incorrectly included vegetation patch; (i) Detected vegetation in the AOI; (j) The original image; (k) AOI with the correctly included vegetation bushes; (l) Detected vegetation in the AOI; (m) The original image; (n) AOI with the correctly included car and incorrectly excluded asphalt; (o) Detected vegetation in the AOI

Problems

- Introducing the preprocessing step did improve accuracy in the area of interest but determining the area is calculation heavy. Is the slight per-pixel accuracy improvement worth the increased calculation time?

- In all methods presented so far, in addition to a trained classifier, we have used additional pre and/or post steps to improve the algorithm. As a long term solution, we did not like this because these additional steps are experimentally determined and we do not have a generalization guarantee.
- From the beginning, our goal was to develop an efficient standalone classifier and that is why, as the next step in our research, we turned to the promising area of deep learning.

3.3.4 Deep learning - Fully Convolutional Neural Network

In this section, we will introduce the basic concepts and the terminology related to Convolutional Neural Networks and present the developed method for vegetation detection.

Convolutional Neural Networks

Convolutional Neural Networks (CNNs, ConvNets) [64, 65, 66] are in many ways similar to the so-called "ordinary" Neural Networks (NN). They are made up out of neurons that have learnable weights and biases. Neurons are organized into layers where the input data represents the input layer, followed by one or more hidden layers and a loss function on the last layer.

Each neuron in an NN applies some function to the input values coming from the receptive field in the previous layer. The function that is applied to the input values is specified by a vector of weights and a bias. These parameters are learned with stochastic gradient descent and computed with backpropagation.

Unlike a regular NN, the CNNs assume that the input is an image. The layers of a CNN have neurons arranged in 3 dimensions: width, height, and depth where the depth corresponds to the third dimension of the image given as input (e.g. RGB image). There are different types of layers that can be used to construct a deep NN. Different layers perform different transformations on their inputs and some are better suited for certain tasks than others. e.g. convolutional layers are likely used for image data [66].

In CNN terminology, the hidden layers, i.e. the convolution layers consist of *filters* which are $n \times n$ matrices design to detect a certain pattern in an image. Filters are also called *kernels* or *feature detectors*. By sliding the filter over the image and computing the dot product forms the so-called *Convolved Feature* or *Activation Map* or the *Feature Map*. Filters act as feature detectors from the original input image. The more filters we have, the more image features get extracted and the better our network becomes at recognizing patterns in unseen images. Images contain multiple types of patterns (edges, textures, shapes, etc.) and these filters are used to detect different types of patterns. For example, an edge detector would be a filter that detects edges. Others are used to detect corners, circles, squares, etc.

When dealing with high-dimensional inputs such as images, it is impractical to connect neurons to all neurons in the previous layer. Instead, each neuron is connected only to a local region of the input volume. This region is called the receptive field of the neuron (equivalently this is the filter size) as shown in Fig 3.23. Each neuron in the convolutional layer is connected only to a local region in the input volume spatially but to the full depth (i.e. all color channels).

There are three main types of layers commonly used to build ConvNet architectures [66]:

- **Convolutional Layer**

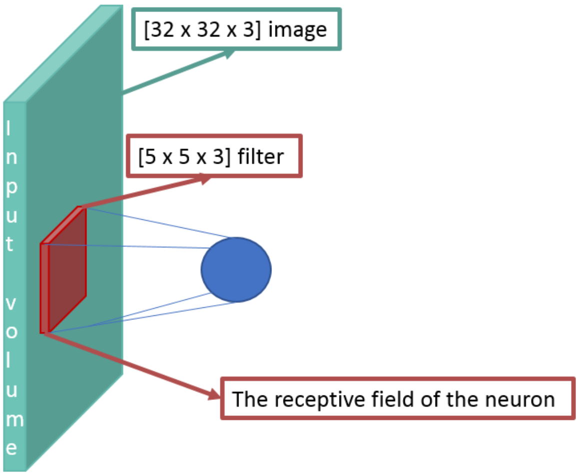


Figure 3.23: A graphical representation of the receptive field of a neuron

It computes the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume.

When working with CNNs the term "volume" is used to describe the layer's inputs and outputs. During the forward pass, each filter is used to slide (convolve) across the width and height of the input volume and compute dot products between the entries of the filter and the input. As a result, we will get a 2-dimensional activation map for every filter used. The activation map gives the responses of a filter at every spatial position. The network will learn filters that activate when they see some type of visual feature such as an edge of some orientation. These activation maps are stacked along the depth dimension and they make up the output volume. This results in an output volume of $[width \times height \times number\_of\_filters]$ dimensions as shown in Fig.3.24.

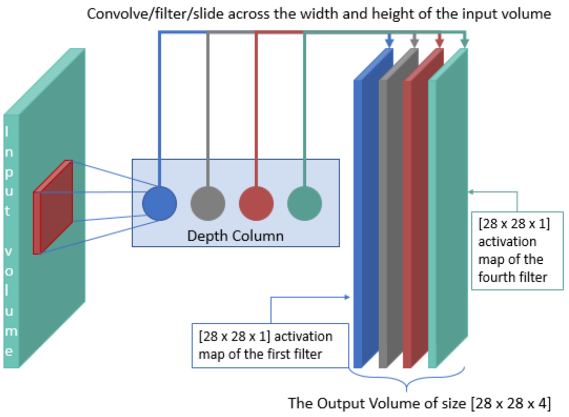

Convolve/filter/slide across the width and height of the input volume
Input volume
Depth Column
[28 x 28 x 1] activation map of the fourth filter
[28 x 28 x 1] activation map of the first filter
The Output Volume of size [28 x 28 x 4]

Figure 3.24: A graphical representation of a filtering sequence and building the output volume of activation maps

There are three hyperparameters that control the size of the output volume:

- **Depth** - corresponds to the number of filters we would like to use, each learning to look for something different in the input. For example, if the first Convolutional Layer takes as input the raw image, then different neurons along the depth dimension may activate in the presence of various oriented edges, or blobs of color. We will refer to a set of neurons that are all looking at the same region of the input as a depth column.

  - **Stride** - the stride with which we slide the filter. When the stride is 1 then the filters move one pixel at a time. When the stride is 2 then the filters jump 2 pixels at a time as they are slid around. This will produce spatially smaller output volumes.
  - **Zero-padding** - sometimes it will be convenient to pad the input volume with zeros around the border. This feature is what allows us to control the spatial size of the output volumes

It is a convention to apply a nonlinear layer (or activation layer) immediately after each convolutional layer. In the past, nonlinear functions like tanh and sigmoid were used, but researchers found out that **ReLU** layers work far better because the network is able to train a lot faster (because of the computational efficiency) without making a significant difference to the accuracy.

ReLU stands for Rectified Linear Unit and is a non-linear operation. Its output is given by:

$$f(x) = max(0,x) \tag{3.9}$$

ReLU is an element wise operation (applied per pixel) and replaces all negative values in the feature map with zero. The purpose of this layer is to introduce nonlinearity to a system that basically has just been computing linear operations during the convolutional layers (element-wise multiplications and summations). Nonlinearity makes it easy for the model to generalize or adapt with a variety of data and to differentiate between the output. It also helps to alleviate the vanishing gradient problem [49, 64]. The vanishing gradient problem is a difficulty found in training artificial neural networks with gradient-based learning methods and backpropagation. In such methods, each of the neural network's weights receives an update proportional to the partial derivative of the error function with respect to the current weight in each iteration of training. The problem is that in some cases, the gradient will be vanishingly small, effectively preventing the weight from changing its value. In the worst case, this may completely stop the neural network from further training.

This layer increases the nonlinear properties of the model and the overall network without affecting the receptive fields of the convolutional layer. The output feature map of this layer is also referred to as the 'Rectified' feature map.

- **Pooling Layer**

It is common to periodically insert a Pooling layer in-between successive Convolutional

layers in a ConvNet architecture. Its function is to progressively reduce the spatial size of the data (downsampling) which reduces the number of parameters and computations in the network. This also controls overfitting.
The Pooling Layer operates independently on every depth slice of the input and resizes it spatially. The most common form of pooling is applying the maximum operation on windows of size 2x2 applied with a stride of 2. This downsamples every depth slice in the input by 2 along both width and height.
Pooling makes the network invariant to small transformations, distortions, and translations in the input image. A small distortion in the input will not change the output of Pooling since we take the maximum value in a local neighborhood.

- **Fully-Connected Layer**
  This layer computes the class scores. The Fully Connected layer is a traditional Multi-Layer Perceptron that uses a softmax activation function in the output layer. The term "Fully Connected" implies that every neuron in the previous layer is connected to every neuron on the next layer.
  The output from the convolutional and pooling layers represent high-level features of the input image. The purpose of the Fully Connected layer is to use these features for classifying the input image into various classes based on the training dataset.

Note that some layers contain parameters and others don't. In particular, the Convolutional and the Fully-Connected layers perform transformations that are a function of not only the activations in the input volume but also of the parameters (the weights and biases of the neurons). On the other hand, the ReLU and the Pooling layers will implement a fixed function. All the parameters are trained with gradient descent so that the class scores that the ConvNet computes are consistent with the labels in the training set for each image.

A **Fully Convolutional neural network (FCN)** is a normal CNN, where the last fully-connected layer is substituted by another convolution layer with a large receptive field, i.e. the last fully-connected layer is converted to a big enough convolutional layer [35]. Fully convolutional indicates that the neural network is composed of convolutional layers without any fully-connected layers. A CNN with fully connected layers is just as end-to-end learnable as a fully convolutional one. The main difference is that the fully convolutional network is learning filters everywhere. Even the decision-making layers at the end of the network are filters.

Vegetation Detection using Fully Convolutional Networks

Our method is inspired by work presented in [35] where authors used deep neural networks for semantic segmentation and scene parsing. Fig. 3.25 illustrates our proposed FCN architecture. The architecture is adapted from the VGG16 architecture [67] which won the ILSVRC14.

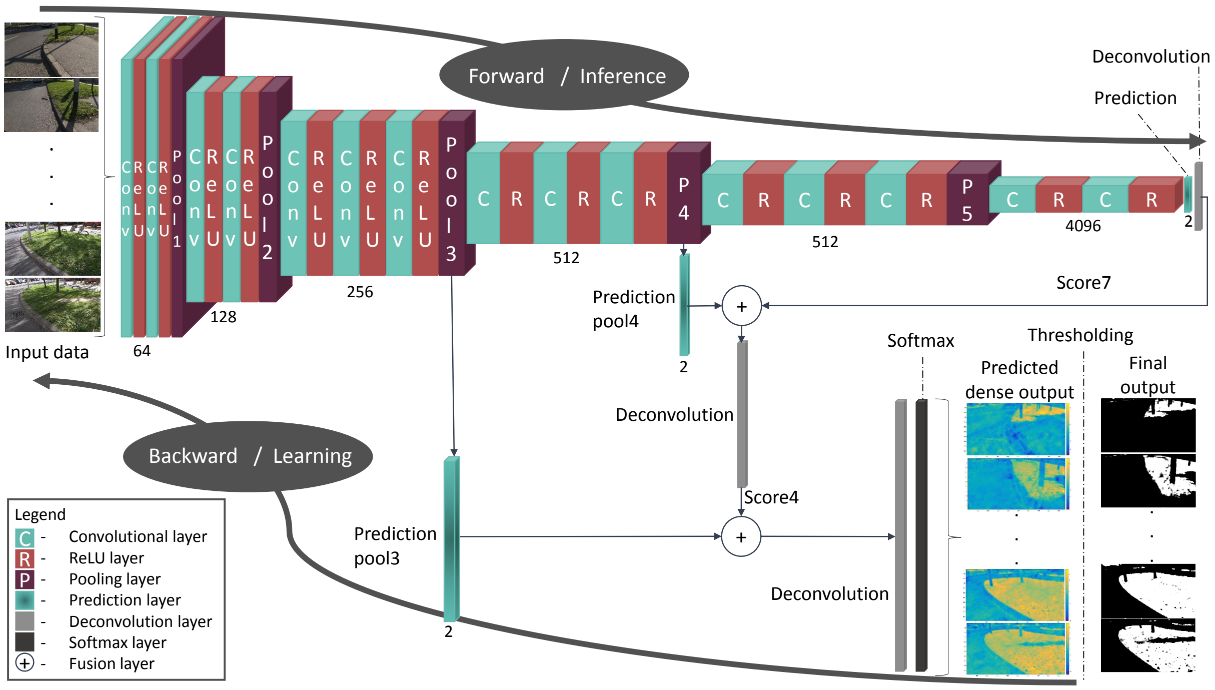


Figure 3.25: A graphical representation of the used FCN for detecting roadside vegetation. As input, the net receives raw RGB images. The net consists of 15 Convolutional layers followed by 15 Rectified Linear Unit activation functions. Max pooling is used to spatially reduce the data size. A Convolutional prediction layer is added at each of the coarse output locations. The prediction layer is followed by a deconvolution layer used to upsample the coarse outputs to the size of the input data. The final result is achieved by appending a Softmax layer at the end of the network followed by thresholding of the dense output to achieve the final result.

The input layer consists of RGB images from our database. Each layer of data in the FCN is a three-dimensional array of size $[h \times w \times d]$, where h and w are the height and width of the images respectively, and d is the feature or channel dimension which, in this case, is three. An FCN naturally operates on an input of any size and produces an output of corresponding spatial dimensions. Both inference and learning are performed whole-image-at-a-time by dense feed-forward computation and back-propagation through the whole net.

The net consists of 15 convolutional layers marked as ’Conv’ or ’C’ in Fig. 3.25 followed by the same number of activation functions marked as ’ReLU’ or ’R’. The numbers under the ’Conv’ and ’ReLU’ layers represents the number of channels of the output data related to that layer, i.e. the size of the output volume.

Max pooling layers are implemented after several ’Conv’ and ’ReLU’ layers to spatially reduce the data size. Subsampling in the layers is necessary to keep filters small and computational requirements reasonable.

The final classification layer from the VGG16 architecture is removed and all, fully connected layers are converted to convolutions. A 1×1 convolution with channel dimension 2 to predict vegetation and non-vegetation (background) in images is added at each of the coarse output locations. The prediction layer is followed by a deconvolution layer used to upsample the coarse outputs to the size of the input data.

The subsampling through the net affects the dimensions of the output which are reduced, and depending on the subsampling factor (stride) the output is more or less coarse and limits the scale of detail in the upsampled output. To address this, *skip steps* are added which combine lower layers with finer strides and the final prediction layer. This is achieved by adding an additional convolution layer on top of ’Pool4’ (in Fig. 3.25 the layer marked as ’Prediction pool4’) to produce additional class predictions. These predictions are fused with the predictions computed on top of the last convolution layer (marked as ’Score7’) by adding a 2x upsampling layer and summing both predictions (this is done in the fusion layer shown in Fig. 3.25). This trend is continued by using the predictions from ’Pool3’ with 2x upsampling of predictions fused from ’Pool4’ and ’Score7’ building the net FCN-8s described in [35].

The networks subsampling layers reduced the size of the image which needs to be restored in order to properly compare the result to the ground truth and to backpropagate the correct values for parameter update in the training process. Some generic upsampling algorithm could be applied to the result to achieve this but in [35] this was solved by adding a learnable upsampling layer which upsamples the feature map in a learnable way. This added *learnable upsampling* layer is called the ’Deconvolution’ layer whose task is to take a low-resolution input and produce a high-resolution output, i.e. to perform upsampling. In the bottom right corner in Fig. 3.25 we see the predicted dense output of the network and the final result achieved by appending a Softmax layer at the end of the network.

The output of the network is an image where each pixel value represents the probability of each pixel being a vegetation pixel. In order to get a binary image, a fixed threshold is applied. The threshold value was calculated by using a Receiver Operating Characteristic (ROC).

Data

The FCN in Fig. 3.25 is trained for segmentation by fine-tuning all layers to learn from whole image inputs and whole image ground truths. The images in our database are recorded in full HD resolution, but in order to keep the training time of the FCN reasonable, we resized our images to 540×960 resolution.

FCN training parameters

Optimization of the network is done by fine-tuning the VGG16 network originally trained for image recognition. All layers of the network are fine-tuned using Stochastic Gradient Descent with the momentum of 0.9. We use a batch size of 20 images, weight decay of 5^{-4}, fixed learning rate of 10^{-4} and all skip layers at once. To properly evaluate how our model would generalize to an independent data set we used 10-fold cross-validation to measure the performance and to validate the results. A single fold of data was trained through 50 epochs.

The new parameters of the prediction and fusion layers acting on ’Pool3’ and ’Pool4’ are zero-initialized so that the net starts with unmodified predictions. Upsampling to the input dimensions is done by deconvolution layers within the net. The final deconvolutional layer is fixed to bilinear interpolation, while intermediate upsampling layers are initialized to bilinear upsampling and then learned.

The dense output of the network was thresholded to get a binary classification image as the final output where the ’ones’ represent the positive class which, in this case, is vegetation and the ’zeros’ represent everything classified as non-vegetation. In order to find an optimal threshold, the True Positive Rate (TPR) and the False Positive Rate (FPR) were calculated for a range of thresholds. By plotting the TPR against the FPR we get the ROC curve for our method. The TPR and FPR were calculated for every image in the database. An example of ROC curves for some of the images is shown in Fig. 3.26.

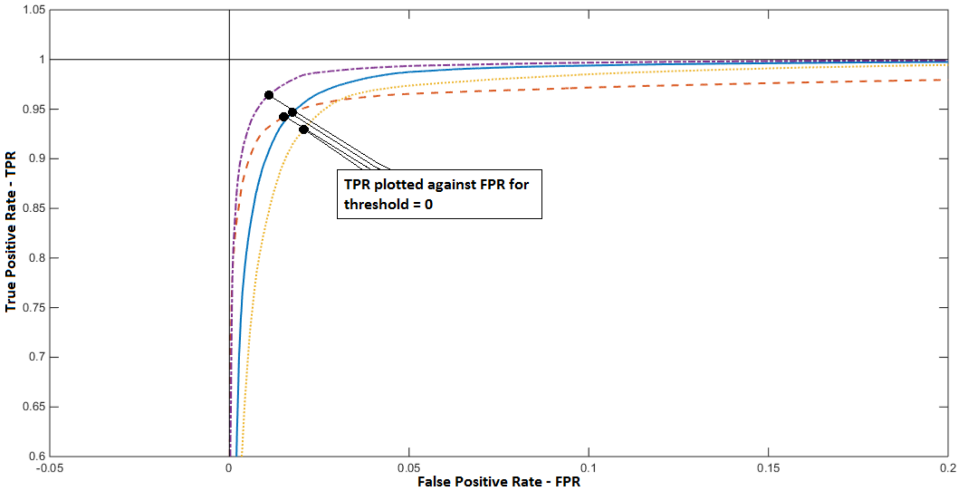


Figure 3.26: The performance of the binary classifier for various threshold values was evaluated by using ROC curves. They were used for determining the optimal threshold for the last layer of the FCN which thresholded the dense output of the network. In this figure, the ROC curve is shown only for some images but a detailed analysis of the whole database was done. Similar results are obtained in a wide range of thresholds, but in this thesis the presented results were obtained with the threshold zero.

Using ROC curves to evaluate the performance of a binary classifier is common and efficient because it shows the balance between true positives and false positives. A random result would be represented by a diagonal line, and a perfect result would be in the error-free point where $FPR = 0$ and $TPR = 1$. For our classifier, similar results are obtained in a wide range of thresholds, which can be seen from the ROC curves in Fig. 3.26 where depending on the image, different thresholds give a better ratio of TPR and FPR (closer to the error-free point). This also indicates that the classifier performance is not sensitive to the chosen threshold. Results presented here were obtained with the threshold zero.

All models were trained and tested using the "Convolutional Neural Networks for MATLAB - MatConvNet" library [68] on NVIDIA GTX 970 graphics card.

Experiments and Results

This method is based on training a classifier using features that are learned from data and not enforced by designers. In order to show the advantages of deep learning techniques in Table 3.3 we compare FCN results to methods based on manually selected features.

The methods presented in Table 3.3 are vegetation detection methods published in [69] and [70]. These methods are comprised of a combination of pre- and post-processing steps which we covered in the previous chapters of this thesis.

The method published in [69] consists of a preprocessing step in which an area of interest is determined using optical flow as described in Chapter 3.3.3. The vegetation detection algorithm is executed inside of the calculated area of interest. With this approach, samples that belong to objects further from the camera which lack in detail the most are excluded from the classification process which in turn decreases the number of samples for classification.

The method published in [70], in addition to the preprocessing step for determining the AOI as in Chapter 3.3.3, it has even a postprocessing step used to correct the output of the classifier by using morphological operations to remove or fill solitary groups of pixels which are presumed to be misclassified.

In addition to these two methods, we also included the results of the method from [70] without any pre- or post-processing steps in order to have a result comparable to the FCN result which also does not have any additional processing steps.

Table 3.3: Comparison of per-pixel accuracy for different vegetation detection techniques.

Method	Remark	Accuracy
[69]	Preprocessing	93.3247%
[70]	Pre- and post-processing	96.10182%
Color + CWT + SVM as described in Chapter 3.3.3	Hand crafted features	93.89%
FCN		96.257%

Fig. 3.27 shows a comparison of VVI and GRVI-based methods to machine learning ones. The binary output in Fig. 3.27(c) and in Fig. 3.27(g) is the result of a conventional machine learning technique based on training an SVM classifier with manually selected features. Both machine learning-based methods (SVM and FCN) gave better results than the VVI and GRVI-based ones. Conventional machine learning misclassified some parts of the problematic objects (Fig. 3.27(c) and Fig. 3.27(g)) which is an improvement compared with the amount of misclassified pixels in Figures 3.27(a), 3.27(e), 3.27(b) and 3.27(f) which are the result of thresholding VVI and GRVI respectively.

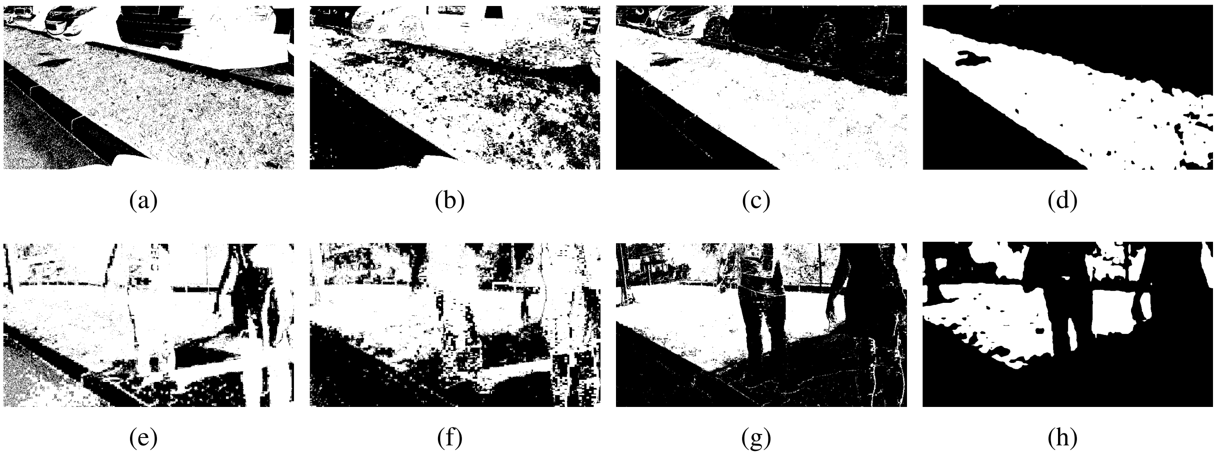


Figure 3.27: Result comparison between simpler methods based on VVI and GRVI and machine learning-based methods on two examples of problematic images. Binary classification result of: (a) VVI, (b) GRVI, (c) SVM, (d) FCN, (e) VVI, (f) GRVI, (g) SVM, (h) FCN

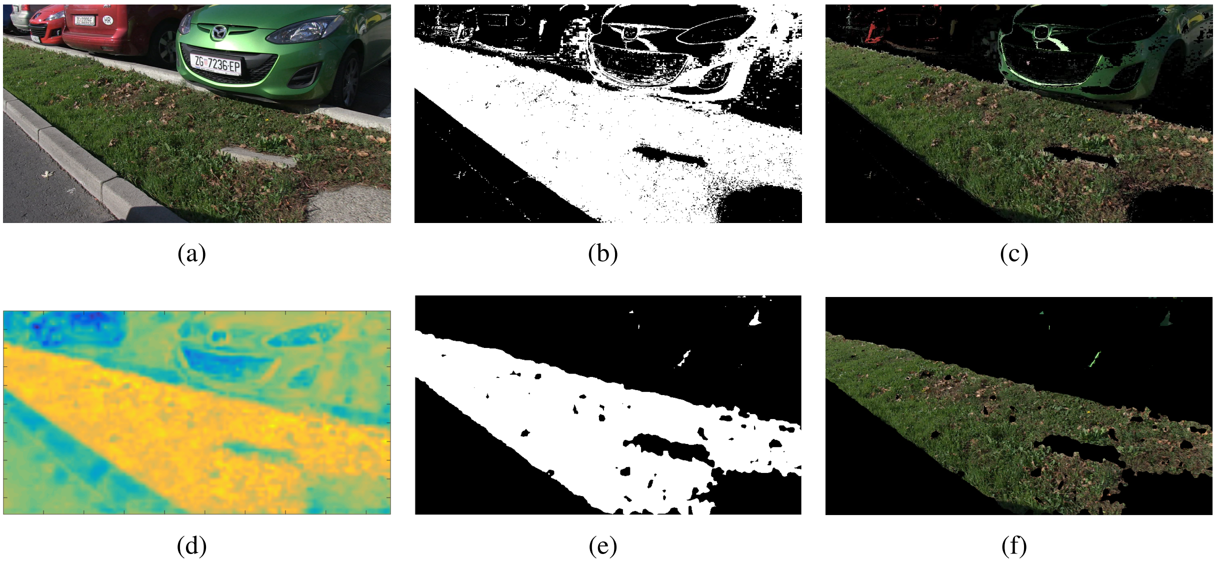
FCN shows further improvement in classification of problematic images compared to the SVM-based technique which can be seen when comparing Fig. 3.27(c) to Fig. 3.27(d) and Fig. 3.27(g) to Fig. 3.27(h). The green objects are correctly classified as non-vegetation. Even in Fig. 3.27(h) the vegetation behind the pedestrians was succesfully detected which shows that a FCN can successfully detect vegetation that is further away from the camera.

The FCN-based method outperforms the SVM-based machine learning method (without pre- and/or post-processing steps) not only in correctly classifying problematic images but in overall per-pixel accuracy. One more example is given in Fig. 3.28. The edges of the green car are falsely detected in Fig. 3.28(c) because they represent high-frequency content. This results in similar values of the calculated texture feature for both the edges and for vegetation. The fully trained deep network did not have this problem. The final output from the network increased the True Negative Rate for this image from 75.56% to 97.6%.

The per-pixel accuracy of the SVM-based method and the method presented in [69] is similar and the small difference is not statistically significant. The features used to train the SVM classifier in all three SVM-based methods are very similar, so the fact that they all achieve similar results is not surprising. The main difference in the feature set is the selection of color features. In [70] the CieLAB color space components were used as color features, while in [69] the color components from multiple color spaces were ranked using feature ranking and the best-ranked color components were used to construct a color feature vector.

The method presented in [70] has higher per-pixel accuracy compared to other SVM-based methods due to an additional postprocessing step which also makes it incomparable to the FCN method.

The advantage of FCN is that it has no need for pre- or post-processing to outperform all other presented methods in every way. Also, raw data is used as input which frees the designers



(a) (b) (c)

(d) (e) (f)

Figure 3.28: Comparison of machine learning method with manually selected features and FCN method with RGB input for a problematic image: (a) Original image, (b) Binary classification result of SVM method, (c) Image visualization of classification result using the SVM method, (d) Predicted Dense Output of FCN, (e) Final result of thresholding the image (d), (f) Image visualization of FCN classification result

of manually selecting features for training.

Finally, the FCN performance was evaluated given a different representation of input data. The model was trained with three different sets of input: RGB, HSV and YUV images. A comparison of per-pixel accuracy is visible in Table 3.4. From Table 3.4 it can be concluded that there is no significant difference between using RGB, HSV or YUV images as input to FCN. This can also be seen from Fig. 3.29 with the example of the problematic image 3.28(a).

Table 3.4: Per-pixel accuracy of FCN performance with three different sets of input: RGB, HSV and YUV images.

Input type	**Accuracy**
FCN + RGB images	96.257%
FCN + HSV images	95.15%
FCN + YUV images	94.53%

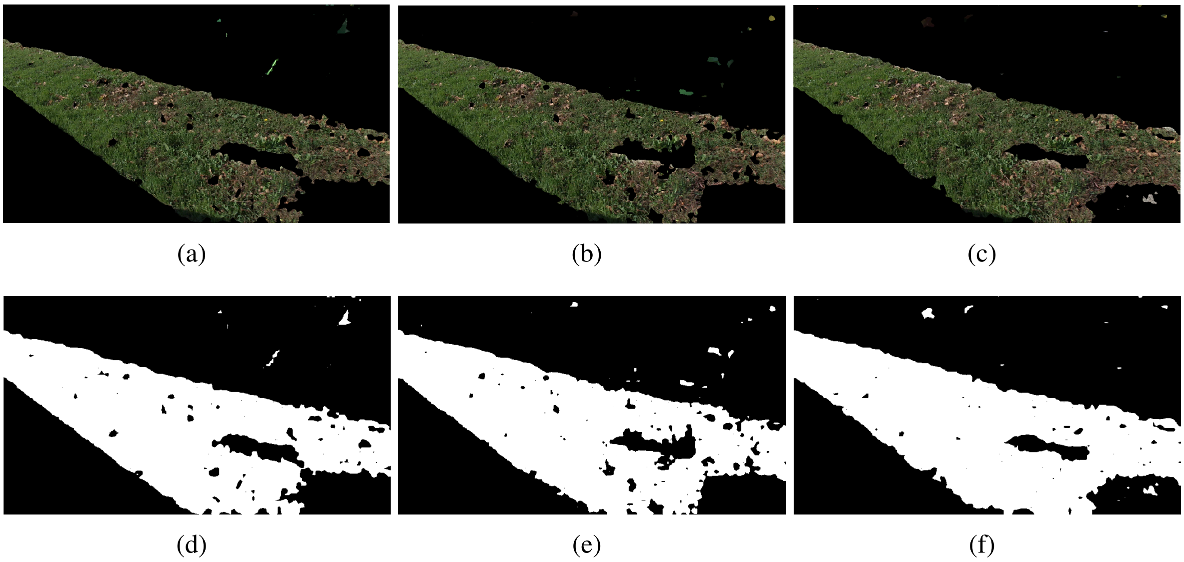


Figure 3.29: Comparison of FCN performance on a problematic image using different color spaces as input data: (a) Result achieved with RGB data, (b) Result achieved with HSV data, (c) Result achieved with YUV data, (d) Binary result achieved with RGB data, (e) Binary result achieved with HSV data, (f) Binary result achieved with YUV data

3.4 Final Discussion and results

Most of the research done in the area of vegetation detection is done on satellite images or other multi-modal recordings. This is a result of previous work done in remote sensing where specific Near Infra Red spectral features were shown to be highly indicative of vegetation. This discovery had an impact on all research that followed which were based on NIR and other modalities invisible to the human eye. Our main focus was on research and development of a vegetation detection method based on using visible spectrum features exclusively. When we started investigating this (some findings were published in 2014.) there was no such research been done by anyone else.

Machine learning techniques are widely used in various image processing applications from segmentation, object detection, recognition, and classification. Such approaches show the best results in these areas so we, also, focused on machine learning techniques while developing our vegetation classification algorithm. Machine learning methods are mostly based on a labeled dataset which is used to train the algorithm to the desired task. In our case, we had to record and label the dataset in order to start training and testing our classifier. We started our research with simpler approaches and in every following method, we addressed some problems we had with the previous one. In the end, we have a comprehensive analysis of the possible issues and we propose solutions for those issues.

Now, when deep learning is gaining more momentum and it is being used in a variety of applications it seemed like a logical continuation of our previous work to investigate this new approach to try and solve our problem more effectively. In the last couple of years some researchers have been investigating the application of various deep learning methods in remote sensing where, again, the algorithms are performed on multispectral data for vegetation detection and monitoring. There is no research done specifically for detecting vegetation using deep learning and color images only (recorded by a simple camera).

The results we accomplished show that deep learning outperforms all other methods and can be used as a tool for detection without using special equipment for NIR recordings. This research represents the first step into a various number of possible applications for vegetation detection including classification of different vegetation types which could have potential usages in habitat mapping, monitoring vegetation not only in urban environments but wherever a vehicle with a camera can go. Our goal, in the beginning, was to provide support for an autonomous vehicle to maintain roadside vegetation, but now, with proof that detection can be done with a significantly high accuracy rate, we can consider other possibilities.

To compare our work with other deep learning-based applications would be difficult since the results obtained in such a way would be hard to interpret. There is no other research done in the specific area of vegetation detection by using only the visible spectrum information based on new methods like Fully Convolutional Networks or other deep learning methods. Object detection and recognition in images and in video is a highly developed area. In contrast, recognizing regions or surfaces in cluttered natural scenes containing multiple object categories (such as cloud, sky, water, vegetation, etc.) is relatively less explored and vegetation detection falls into that category. That is why all research done in object recognition compared to our findings would not be statistically significant.

It should be made clear that many publications mentioning successful vegetation detection often indicate some very specific species of vegetation or a very limited dataset. And these approaches are then just used for robots operating in a specific environment or for satellite images. They are not used for roadside vegetation detection using a camera mounted on a vehicle. Due to different goals, those methods cannot be directly compared to our method for detection of vegetation.

In Table 3.5 we summarize the results presented throughout this thesis. We present the accuracy achieved only by the classifier and excluding the results achieved with the pre- and/or post-processing steps. Since we had a specific application in mind we added these steps in order to develop a better and more optimal solution for such usage. These algorithms were published and their performance was addressed in the previous chapter (Chapter 3.3.4).
Here, we want to focus only on the performance of the classifiers.

Table 3.5: Vegetation detection results summary.

Feature vector	**Accuracy**	**Feature vector**	**Accuracy**
CieLAB	91,9015%	AB	89,9947%
YUV	91,8235%	UV	61,7462%
HSV	91,4285%	HS	88,7151%
RGB	89,6314%		
VVI	58,342%		
GRVI	67,617%		
Color + Entropy	92,436%		
Color + CWT	93,89%		
FCN	96,257%		

Even though vegetation indices are the number one metric for vegetation detection in remote sensing, the same does not apply for indices based on the visible spectrum values. When working with RGB values the results are sensitive to changes in luminance due to its additive nature which makes working with this color space not so robust. Analysis of results obtained by calculating and thresholding these indices showed that this kind of approach is not usable in an on-ground scenario. Even using machine learning methods and training classifiers with color features from different color spaces did not yield satisfactory results. However, we did draw some conclusions from these experiments:

- VVI and GRVI performed with lowest classification accuracy which is not surprising due to lack of information in these vegetation indices
- Slightly better performance is achieved when using the lightness component together with the color components
- CieLAB, YUV, and HSV color spaces perform similarly and are giving slightly higher accuracy results compared to RGB which was expected due to the fact that every component contains luminance information which makes it more sensitive to luminance changes

Using only color features resulted in false positive detections of objects similar to vegetation in color which is a problem, especially if the intended use of the algorithm is to aid in roadside maintenance where it is important to know if, for example, a green car is not vegetation. We expanded the feature vector with texture features in order to differentiate these false positives from vegetation. First, we used a simple feature to describe texture - entropy, followed by a more complex one based on CWT. Both approaches showed improvement in correctly classifying parts of the problematic objects, even though the overall per-pixel accuracy improved only slightly. That is why we put more focus on these problematic images/scenes and developed further improvements with the goal to minimize these issues and not just paying attention to the overall per-pixel accuracy. CWT as a texture descriptor has shown improvements in these problematic images compared to using only color or using color with entropy. CWT is powerful due to its robustness to orientation and scale which for vegetation means a lot since in real-world applications one can expect blades of grass to go in all sorts of directions, and different types of vegetation will have leaves of different sizes so the scale invariance also makes a difference.

We also presented some possible pre- and post-processing steps which were tested in order to improve the overall detection method, but for every improvement, there was a compromise (e.g. speed) and that was something we wanted to avoid.

In the final step of our research, we looked for a possible solution that would not depend on any additional steps in order to achieve satisfactory performance, especially for the problematic images we mentioned. FCN proved to be the solution we were looking for. FCN outperformed all other methods regardless of the additional steps used. Even the True Negative Rate increased significantly which was more important for our intended usage then the per-pixel accuracy which was also significantly higher than in other cases.

Training a classifier using features that are learned from data and not enforced by designers proved to give the best results for detecting roadside vegetation.

Chapter 4

Conclusion

Most of the research done in the area of vegetation detection is done on satellite images or other multi-modal recordings. This is a result of previous work done in remote sensing where specific Near Infra Red spectral features were shown to be highly indicative of vegetation. This discovery had an impact on all research that followed which were based on NIR and other modalities invisible to the human eye.

Our main focus was research and development of a vegetation detection method using features from the visible spectrum exclusively. Our goal was to provide support for an autonomous vehicle in roadside maintenance, but once vegetation is detected it can further be classified which would provide information about different types of vegetation not only in urban areas but wherever a vehicle with a camera can go.

Detecting roadside vegetation is a specific application for which we had to record a database of images containing roadside vegetation in various conditions. Creating the database and labeling the recorded data was the first step in our research. Once we had the data we decided to use machine learning to train a classifier for differentiating vegetation from non-vegetation.

The first developed method was based on using only color features for training an SVM classifier. The goal was to determine if there is a difference in detection accuracy when using different color spaces or does the classifier perform better when we exclude the luminance component since that is one of the biggest issues when working with real-life recordings.

We concluded that using the lightness component together with color performed slightly better than without it and that from all four color spaces evaluated CieLAB, YUV and HSV performed similarly and they all resulted in slightly higher per-pixel classification accuracy compared to RGB. In the end, CieLAB had the highest accuracy of $91,9015\%$ and it was chosen as the color feature in the following methods.

To solve the issues of falsely detecting green (and other vegetation-like colors) objects as vegetation we added a new feature for describing texture. We started with a simple texture feature - entropy which is described as a measure of disorder in a system. The idea was that vegetation areas will have a bigger amount of disorder, compared to cars, pedestrians, asphalt and other roadside objects. This approach did not improve the per-pixel accuracy drastically ($92,436\%$), but it did manage to correctly classify some objects or parts of objects similar to vegetation in color. Here, we added a postprocessing step in order to improve detection boundaries, but we were not satisfied with the performance of the added texture feature because some parts of problematic objects where the entropy is high are still falsely detected as vegetation.

In order to improve these results we had to add a more complex feature that would describe texture and be scale and rotation invariant. Using CWT in a combination with feature ranking we calculated a texture feature that was determined to contain the most significant information. The SVM classifier trained with such features resulted in an slight increase in per-pixel accuracy ($93,89\%$) and in a slight improvement when classifying problematic images.
In the end, the trained classifier represented one part of a bigger pipeline that consisted of additional pre- and/or post-processing steps. The classifiers results were improved ($96,10182\%$) due to these additional steps, but we were not satisfied because these additional steps increased the methods complexity which resulted in higher training and classification times.

Now, when deep learning is gaining more momentum and is being used in a variety of applications it seemed like a logical continuation of our previous work to investigate this approach and try to solve our problem more effectively. In the last couple of years some researchers have been investigating the application of various deep learning methods in remote sensing where, again, the algorithms are performed on multispectral data for vegetation detection and monitoring. There is no research done specifically for detecting vegetation using deep learning and color images only (recorded by a simple camera).

The results we accomplished with a trained Fully Convolutional Network show that deep learning outperforms all other methods and can be used as a tool for detection without using special equipment for NIR recordings. We have compared the performance of the proposed method with other published methods for roadside vegetation detection and the results show that the FCN outperforms all other presented methods in per-pixel accuracy and that the performance of the proposed method in problematic cases is where its advantage is most evident.

This research represents the first step into a various number of possible applications for vegetation detection including classification of different vegetation types which could have potential usages for detecting and mapping habitats, monitoring vegetation not only in urban environments but wherever a vehicle with a camera can go. Our goal, in the beginning, was to provide support for an autonomous vehicle to maintain roadside vegetation, but now, with proof that detection can be done with a significantly high accuracy rate, we can consider other possibilities.

The results achieved provides us with many possibilities for future work where the vegetation detection could just be the first step in a bigger system.

Bibliography

[1] Carranza, E. J. M., Hale, M., “Remote detection of vegetation stress for mineral exploration”, in Geoscience and Remote Sensing Symposium, 2001. IGARSS ’01. IEEE 2001 International, Vol. 3, 2001, str. 1324-1326 vol.3.

[2] Pan, Z., Wang, F., Xia, L., Wang, X., “Feature extraction for urban vegetation stress identification using hyperspectral remote sensing”, in Information Science and Engineering (ICISE), 2010 2nd International Conference on, 2010, str. 250-254.

[3] Arnold, T., Biasio, M. D., Fritz, A., Leitner, R., “Uav-based measurement of vegetation indices for environmental monitoring”, in 2013 Seventh International Conference on Sensing Technology (ICST), Dec 2013, str. 704-707.

[4] Ghazal, M., Khalil, Y. A., Hajjdiab, H., “Uav-based remote sensing for vegetation cover estimation using ndvi imagery and level sets method”, in 2015 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT), Dec 2015, str. 332-337.

[5] Zhang, X., Song, Y., Wang, S., Zhang, L., Zhang, X., “A method for extracting vegetation information of urban underlaying surface oriented to eco-environmental quality assessment”, in 2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), July 2017, str. 3479-3486.

[6] Langford, Z. L., Kumar, J., Hoffman, F. M., “Convolutional neural network approach for mapping arctic vegetation using multi-sensor remote sensing fusion”, in 2017 IEEE International Conference on Data Mining Workshops (ICDMW), Nov 2017, str. 322-331.

[7] Zhan, Z., Lai, B., “Vegetation detection of close-range images for landslide monitoring”, in Computer Vision in Remote Sensing (CVRS), 2012 International Conference on, 2012, str. 13-18.

[8] Chowdhury, S., Verma, B., “A novel feature extraction technique to retrieve vegetation class for fire risk assessment”, in Signal Processing and Communication Systems (ICSPCS), 2014 8th International Conference on, Dec 2014, str. 1-6.

[9] Chowdhury, S., Verma, B., Stockwell, D., “A novel texture feature based multiple classifier technique for roadside vegetation classification”, Expert Systems with Applications, Vol. 42, No. 12, 2015, str. 5047 - 5055, dostupno na: http://www.sciencedirect.com/science/article/pii/S0957417415001542

[10] Bradley, D., Unnikrishnan, R., Bagnell, J., “Vegetation detection for driving in complex environments”, in Robotics and Automation, 2007 IEEE International Conference on, 2007, str. 503-508.

[11] Da-xue, L., Tao, W., Bin, D., “Fusing ladar and color image for detection grass off-road scenario”, in Vehicular Electronics and Safety, 2007. ICVES. IEEE International Conference on, 2007, str. 1-4.

[12] Nguyen, D. V., Kuhnert, L., Kuhnert, K. D., “Spreading algorithm for efficient vegetation detection in cluttered outdoor environments”, Robotics and Autonomous Systems, Vol. 60, No. 12, 2012, str. 1498 - 1507.

[13] Nguyen, D. V., Kuhnert, L., Jiang, T., Thamke, S., Kuhnert, K. D., “Vegetation detection for outdoor automobile guidance”, in Industrial Technology (ICIT), 2011 IEEE International Conference on, 2011, str. 358-364.

[14] Nguyen, D. V., Kuhnert, L., Thamke, S., Schlemper, J., Kuhnert, K. D., “A novel approach for a double-check of passable vegetation detection in autonomous ground vehicles”, in Intelligent Transportation Systems (ITSC), 15th International IEEE Conference on, 2012, str. 230-236.

[15] Nguyen, D. V., Kuhnert, L., Kuhnert, K. D., “Structure overview of vegetation detection. A novel approach for efficient vegetation detection using an active lighting system”, Robotics and Autonomous Systems, Vol. 60, No. 4, 2012, str. 498 - 508.

[16] Bengio, Y., “Deep learning of representations for unsupervised and transfer learning”, in Proceedings of ICML Workshop on Unsupervised and Transfer Learning, ser. Proceedings of Machine Learning Research, Guyon, I., Dror, G., Lemaire, V., Taylor, G., Silver, D., (ur.), Vol. 27. Bellevue, Washington, USA: PMLR, 02 Jul 2012, str. 17–36, dostupno na: http://proceedings.mlr.press/v27/bengio12a.html

[17] Zafarifar, B., de With, P., “Grass field detection for TV picture quality enhancement”, in Consumer Electronics, 2008. ICCE 2008. Digest of Technical Papers. International Conference on, 2008, str. 1-2.

[18] Herman, S., Janssen, J., Bellers, E., Wendorf, J., “Automatic segmentation-based grass detection for real-time video”, uS Patent 6,832,000. Dec. 14 2004.

[19] The Planetary Habitability Laboratory, “Planetary habitability laboratory”, http://phl.upr.edu/projects/visible-vegetation-index-vvi, 2014.

[20] Takeshi, M., Nasahara, K. N., Oguma, H., Tsuchida, T., “Applicability of green-red vegetation index for remote sensing of vegetation phenology”, Remote Sensing, Vol. 2, 1/2010 2010, str. 2369 - 2387.

[21] Fan, H., Mei, X., Prokhorov, D., Ling, H., “Cross datasets vegetation detection with spatial prior and local context”, in 2016 IEEE Intelligent Vehicles Symposium (IV), June 2016, str. 735-740.

[22] Jordan, C. F., “Derivation of leaf-area index from quality of light on the forest floor”, Ecology, Vol. 50, No. 4, 1969, str. 663-666, dostupno na: https://esajournals.onlinelibrary.wiley.com/doi/abs/10.2307/1936256

[23] Barroso, C., Monteiro, I., “Monitoring vegetation from space”, http://www.eumetrain.org/data/3/36/print.htm, 2010.

[24] Krizhevsky, A., Sutskever, I., Hinton, G. E., “Imagenet classification with deep convolutional neural networks”, in Advances in Neural Information Processing Systems 25, Pereira, F., Burges, C. J. C., Bottou, L., Weinberger, K. Q., (ur.). Curran Associates, Inc., 2012, str. 1097–1105.

[25] Simonyan, K., Zisserman, A., “Very deep convolutional networks for large-scale image recognition”, CoRR, Vol. abs/1409.1556, 2014.

[26] Li Deng, B. K., Geoffrey Hinton, “New types of deep neural network learning for speech recognition and related applications: An overview”, in IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), May 2013, May 2013.

[27] Ciresan, D., Giusti, A., Gambardella, L. M., Schmidhuber, J., “Deep neural networks segment neuronal membranes in electron microscopy images”, in Advances in Neural Information Processing Systems 25, Pereira, F., Burges, C. J. C., Bottou, L., Weinberger, K. Q., (ur.). Curran Associates, Inc., 2012, str. 2843–2851.

[28] Farabet, C., Couprie, C., Najman, L., LeCun, Y., “Learning hierarchical features for scene labeling”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 35, No. 8, Aug 2013, str. 1915-1929.

[29] Ganin, Y., Lempitsky, V. S., “N^4-fields: Neural network nearest neighbor fields for image transforms”, CoRR, Vol. abs/1406.6558, 2014, dostupno na: http://arxiv.org/abs/1406.6558

[30] Gupta, S., Girshick, R. B., Arbelaez, P., Malik, J., “Learning rich features from RGB-D images for object detection and segmentation”, CoRR, Vol. abs/1407.5736, 2014, dostupno na: http://arxiv.org/abs/1407.5736

[31] Hariharan, B., Arbeláez, P., Girshick, R., Malik, J., “Simultaneous detection and segmentation”, in European Conference on Computer Vision (ECCV), 2014.

[32] Pinheiro, P. H. O., Collobert, R., “Recurrent convolutional neural networks for scene parsing”, CoRR, Vol. abs/1306.2795, 2013, dostupno na: http://arxiv.org/abs/1306.2795

[33] dos Santos, C. N., Gatti, M., “Deep convolutional neural networks for sentiment analysis of short texts.”, in COLING, 2014, str. 69–78.

[34] Ciresan, D. C., Meier, U., Gambardella, L. M., Schmidhuber, J., “Convolutional neural network committees for handwritten character classification”, in 2011 International Conference on Document Analysis and Recognition, Sept 2011, str. 1135-1139.

[35] Shelhamer, E., Long, J., Darrell, T., “Fully convolutional networks for semantic segmentation”, CoRR, Vol. abs/1605.06211, 2016, dostupno na: http://arxiv.org/abs/1605.06211

[36] Audebert, N., Saux, B. L., Lefèvre, S., “Semantic segmentation of earth observation data using multimodal and multi-scale deep networks”, CoRR, Vol. abs/1609.06846, 2016, dostupno na: http://arxiv.org/abs/1609.06846

[37] Valada, A., Oliveira, G., T.Brox, Burgard, W., “Robust semantic segmentation using deep fusion”, in Robotics: Science and Systems (RSS 2016) Workshop, Are the Sceptics Right? Limits and Potentials of Deep Learning in Robotics, 2016, dostupno na: http://lmb.informatik.uni-freiburg.de/Publications/2016/OB16d

[38] Potena, C., Nardi, D., Pretto, A., “Fast and accurate crop and weed identification with summarized train sets for precision agriculture”, in Intelligent Autonomous Systems 14, Chen, W., Hosoda, K., Menegatti, E., Shimizu, M., Wang, H., (ur.). Cham: Springer International Publishing, 2017, str. 105–121.

[39] Suarez, P. L., Sappa, A. D., Vintimilla, B. X., “Learning image vegetation index through a conditional generative adversarial network”, in 2017 IEEE Second Ecuador Technical Chapters Meeting (ETCM), Oct 2017, str. 1-6.

[40] Jin, X., Han, J., K-Means Clustering. Boston, MA: Springer US, 2010, str. 563–564, dostupno na: https://doi.org/10.1007/978-0-387-30164-8_425

[41] Krejcie, R. V., Morgan, D. W., “Determining sample size for research activities”, Educational and psychological measurement, Vol. 30, No. 3, 1970, str. 607–610.

[42] Guyon, I., Elisseeff, A., “An introduction to variable and feature selection”, J. Mach. Learn. Res., Vol. 3, Mar. 2003, str. 1157–1182.

[43] Japkowicz, N., Shah, M., Evaluating Learning Algorithms: A Classification Perspective. Cambridge University Press, 2011.

[44] Burges, C. J., “A tutorial on support vector machines for pattern recognition”, Data Mining and Knowledge Discovery, Vol. 2, No. 2, Jun 1998, str. 121–167, dostupno na: https://doi.org/10.1023/A:1009715923555

[45] Hofmann, M., “Support vector machines — kernels and the kernel trick”, Houptseminar report 2006, 2006.

[46] Mitchell, T. M., Machine Learning, 1st ed. New York, NY, USA: McGraw-Hill, Inc., 1997.

[47] Goodfellow, I., Bengio, Y., Courville, A., Deep Learning. MIT Press, 2016, http://www.deeplearningbook.org.

[48] LeCun, Y., Bengio, Y., Hinton, G. E., “Deep learning”, Nature, Vol. 521, No. 7553, 2015, str. 436–444, dostupno na: https://doi.org/10.1038/nature14539

[49] Nielsen, M. A., “Neural networks and deep learning”, dostupno na: http://neuralnetworksanddeeplearning.com/ 2018.

[50] Tesauro, G., “Practical issues in temporal difference learning”, Mach. Learn., Vol. 8, No. 3-4, May 1992, str. 257–277, dostupno na: https://doi.org/10.1007/BF00992697

[51] Hinton, G. E., Osindero, S., Teh, Y.-W., “A fast learning algorithm for deep belief nets”, Neural Comput., Vol. 18, No. 7, Jul. 2006, str. 1527–1554, dostupno na: http://dx.doi.org/10.1162/neco.2006.18.7.1527

[52] Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H., “Greedy layer-wise training of deep networks”, in Advances in Neural Information Processing Systems 19, Schölkopf, B., Platt, J. C., Hoffman, T., (ur.). MIT Press, 2007, str. 153–160, dostupno na: http://papers.nips.cc/paper/3048-greedy-layer-wise-training-of-deep-networks.pdf

[53] Dougherty, E., An introduction to morphological image processing, ser. Tutorial texts in optical engineering. SPIE Optical Engineering Press, 1992.

[54] Kachigan, S., Multivariate Statistical Analysis: A Conceptual Introduction. Radius Press, 1991, dostupno na: https://books.google.hr/books?id=eJhpAAAAMAAJ

[55] Gonzalez, R. C., Woods, R. E., Eddins, S. L., Digital Image Processing Using MATLAB. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2003.

[56] Antoine, J. P., “The continuous wavelet transform in image processing”, BCWI Quarterly, Vol. 11, No. 4, 1998, str. 323-345.

[57] Saxena, A., Chung, S. H., Ng, A. Y., “Learning depth from single monocular images”, in In NIPS 18. MIT Press, 2005.

[58] Wedel, A., Franke, U., Klappstein, J., Brox, T., Cremers, D., “Realtime depth estimation and obstacle detection from monocular video”, in Pattern Recognition (Proc. DAGM), ser. LNCS, et al., K. F., (ur.), Vol. 4174. Berlin, Germany: Springer, September 2006, str. 475–484.

[59] Horn, B. K., Schunck, B. G., “Determining optical flow”, Cambridge, MA, USA, Tech. Rep., 1980.

[60] Lucas, B. D., Kanade, T., “Optical navigation by the method of differences”, in Proceedings of the 9th International Joint Conference on Artificial Intelligence. Los Angeles, CA, USA, August 1985, 1985, str. 981–984, dostupno na: http://ijcai.org/Proceedings/85-2/Papers/060.pdf

[61] Liu, C., Beyond Pixels: Exploring New Representations and Applications for Motion Analysis. MIT, 2009.

[62] Brox, T., Bruhn, A., Papenberg, N., Weickert, J., “High accuracy optical flow estimation based on a theory for warping”, Pajdla, T., Matas, J., (ur.), Vol. 3024. Prague, Czech Republic: Springer, May 2004, str. 25–36.

[63] Bruhn, A., Weickert, J., Schnörr, C., “Lucas/kanade meets horn/schunck: Combining local and global optic flow methods”, Int. J. Comput. Vision, Vol. 61, No. 3, Feb. 2005, str. 211–231.

[64] O’Shea, K., Nash, R., “An introduction to convolutional neural networks”, ArXiv e-prints, 11 2015.

[65] Krizhevsky, A., Sutskever, I., Hinton, G. E., “Imagenet classification with deep convolutional neural networks”, in Advances in Neural Information Processing Systems 25, Pereira, F., Burges, C. J. C., Bottou, L., Weinberger, K. Q., (ur.). Curran Associates, Inc., 2012, str. 1097–1105, dostupno na: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[66] Karpathy, A., “Cs231n convolutional neural networks for visual recognition”, http://cs231n.github.io/convolutional-networks/, 2017.

[67] Simonyan, K., Zisserman, A., “Very deep convolutional networks for large-scale image recognition”, CoRR, Vol. abs/1409.1556, 2014, dostupno na: http://arxiv.org/abs/1409.1556

[68] Vedaldi, A., Lenc, K., “Matconvnet – convolutional neural networks for matlab”, in Proceeding of the ACM Int. Conf. on Multimedia, 2015.

[69] Harbaš, I., Subašić, M., “Cwt-based detection of roadside vegetation aided by motion estimation”, in Visual Information Processing (EUVIP), 2014 5th European Workshop on, Dec 2014, str. 1-6.

[70] Harbaš, I., Subašić, M., “Motion estimation aided detection of roadside vegetation”, in Image and Signal Processing (CISP), 2014 7th International Congress on, Oct 2014, str. 420-425.

Biography

Iva Harbaš was born on 22nd of April in 1988. in Cazin, Bosnia and Herzegovina. There, she finished high school after which, in 2007. she enrolled at the Faculty of Elctrical Engineering and Computing in Zagreb.

Iva attended the Faculty from 2007. to 2012. During this time she got her Bachelor of Science in Computing degree in the year 2010. where she specialized in Information Processing and Multimedia Systems. In 2012. she finished the study program Information and Communication Technology which awarded her with a Master of Science in Information and Communication Technology degree. She graduated Magna cum Laude.

That same year she enrolled in the postgraduate doctoral study of Computing where she started her PhD education. In the following year, Iva started working on an European IPA IIIc project at her alma mater. For two years she worked as a part of a team researching and developing computer vision solutions for safe traffic from where she got the topic for her PhD thesis. During this time her work was mostly oriented in the field of computer vision, image processing and machine learning.

In 2015., after the project finished Iva worked in a startup company after which she moved to Ericsson Nikola Tesla, Zagreb where she is currently employed as a software developer.

List of published papers

Paper in Journal

1. Harbaš, Iva; Prentašić, Pavle; Subašić, Marko. Detection of roadside vegetation using Fully Convolutional Networks. Image and Vision Computing. 74 (2018); 1-9 (journal article).

Papers on International Conferences

1. Harbaš, Iva; Subašić, Marko. "Detection of Roadside Vegetation Using Features from the Visible Spectrum", Mipro 2014 proceedings, Biljanović, Petar, editor(s). Opatija, Croatia, 2014. 1454-1459 (lecture,international peer-review,published,scientific).

2. Harbaš, Iva; Subašić, Marko. "CWT-based Detection of Roadside Vegetation Aided by Motion Estimation", Visual Information Processing (EUVIP), 2014 5th European Workshop on. Paris, 2014. (poster,international peer-review,published,scientific).
3. Harbaš, Iva; Subašić, Marko. "Motion Estimation Aided Detection of Roadside Vegetation", Image and Signal Processing (CISP), 2014 7th International Congress on. Dalian, 2014. (lecture,international peer-review,published,scientific).
4. Harbaš, Iva; Banić, Nikola; Jurić, Darko; Lončarić, Sven; Subašić, Marko. "Computer vision-based advanced driver assistance systems", Proceedings of KoREMA. Šakić, Željko, editor(s). Zagreb, 2014. 17-20 (lecture,international peer-review,published,expert).

Životopis

Iva Harbaš je rođena 22. travnja 1988. godine u Cazinu, Bosna i Hercegovina. Tamo je završila srednju školu, a 2007. godine upisuje Fakultet elektrotehnike i računarstva u Zagrebu.

Pohađala je FER od 2007. do 2012. godine. Za to vrijeme stekla je diplomu sveučilišnog prvostupnika inženjera računarstva u 2010. godini, gdje se specijalizirala u području obrade informacija i multimedijskih sustava. Nakon toga, 2012. godine je završila studijski program 'Informacijska i komunikacijska tehnologija' i stekla zvanje magistra inženjera informacijskih i komunikacijskih tehnologija. Diplomirala je Magna cum Laude.

Iste godine upisala je poslijediplomski doktorski studij računarstva gdje je započela doktorski studij. Sljedeće godine, Iva je počela raditi na IPA IIIc Europskom projektu gdje je dvije godine radila kao dio tima koji je istraživao i razvijao rješenja za siguran promet bazirana na računalnom vidu odakle je dobila temu za doktorsku disertaciju. Tijekom tog vremena bavila se istraživanjem i razvojem u području računalnog vida, obrade slika i strojnog učenja.

U 2015. godini, nakon završetka projekta, godinu dana je radila u istraživačkom timu jednog startupa, nakon čega se zaposlila u Ericsson Nikola Tesla u Zagrebu gdje je trenutno zaposlena.