Variational methods for restoration of images degraded by fog and rain

Stipetić, Vedran

University of Zagreb

FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

Vedran Stipetić

VARIATIONAL METHODS FOR RESTORATION OF IMAGES DEGRADED BY FOG AND RAIN

DOCTORAL THESIS

Zagreb, 2023

University of Zagreb

FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

Vedran Stipetić

VARIATIONAL METHODS FOR RESTORATION OF IMAGES DEGRADED BY FOG AND RAIN

DOCTORAL THESIS

Supervisor: Academic Professor Sven Lončarić, F.C.A

Zagreb, 2023

Sveučilište u Zagrebu

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Vedran Stipetić

VARIJACIJSKE METODE ZA OBNAVLJANJE SLIKA DEGRADIRANIH MAGLOM I KIŠOM

DOKTORSKI RAD

Mentor: akademik prof. dr. sc. Sven Lončarić

Zagreb, 2023.

This doctoral thesis was completed at the University of Zagreb Faculty of Electrical Engineering and Computing, Department of Electronic Systems and Information Processing. This research has been supported by the Croatian Science Foundation under the grant DOK-09-2018-9742.

Supervisor: Academic Professor Sven Lončarić, F.C.A

The dissertation has: 88 pages

Dissertation number: _____

O mentoru

Sven Lončarić diplomirao je i magistrirao u polju elektrotehnike na Fakultetu elektrotehnike i računarstva, 1985. i 1989. godine. Doktorirao je u polju elektrotehnike na Sveučilištu u Cincinnatiju, SAD, 1994. godine. U zvanje redoviti profesor u trajnom zvanju u polju elektrotehnike i polju računarstva na FER-u izabran je 2011. godine. Bio je suradnik ili voditelj na brojnim istraživačkim i razvojnim projektima u području razvoja metoda za obradu slika i računalnog vida. Od 2001. do 2003. bio je Assistant Professor na Sveučilištu New Jersey Institute of Technology, SAD. Voditelj je istraživačkog laboratorija za obradu slike na FER-u. Osnivač je i voditelj Centra izvrsnosti za računalni vid na Sveučilištu u Zagrebu. Suvoditelj je nacionalnog Znanstvenog centra izvrsnosti za znanost o podatcima i kooperativne sustave i voditelj Centra za umjetnu inteligenciju FER-a. Sa svojim studentima i suradnicima publicirao je više od 250 znanstvenih i stručnih radova. Prof. Lončarić redoviti je član Hrvatske akademije znanosti i umjetnosti. Prema studiji Sveučilišta Stanford objavljenoj 2022. godine rangiran je u 2% najutjecajnijih svjetskih znanstvenika u kategoriji umjetne inteligencije i obrade slike. Za svoj znanstveni i stručni rad dobio je više nagrada uključujući Državnu nagradu za znanost.

About the Supervisor

Sven Lončarić received Diploma of Engineering and Master of Science degrees in electrical engineering from the Faculty of Electrical Engineering and Computing in 1985 and 1989, respectively. He received a Ph.D. degree in electrical engineering from University of Cincinnati, USA, in 1994. Since 2011, he has been a tenured full professor in electrical engineering and computer science at FER. He was a project leader on a number of research projects in the area of image processing and computer vision. From 2001-2003, he was an assistant professor at New Jersey Institute of Technology, USA. He founded the Image Processing Laboratory at FER and the Center for Computer Vision at University of Zagreb. Prof. Lončarić has been a co-director of the national Center of Research Excellence in Data Science and Cooperative Systems and the director of the Center for Artificial Intelligence at FER. With his students and collaborators he has published more than 250 scientific papers. Prof. Lončarić is a full member of the Croatian Academy of Sciences and Arts. According to a Stanford University study published in 2022 he was ranked in the top 2% of the most cited world scientists in the category artificial intelligence – image processing. For his scientific work he received several awards including the National Science Award.

Ackownledgements

There are many people I owe gratitude for helping me finish this doctoral thesis. First and foremost is, of course, my supervisor Academic Professor Sven Lončarić. Thank you for guiding me from the very start of my journey in computer vision to finishing a Ph.D. in the field. I would also like to thank my family, who supported and helped me in my studies however they could. A huge thank you to my friends, with whom I've had many late-night conversations about mathematics and computer science, and especially to Vlatka, who found at least a hundred spelling and grammar errors in this thesis. And finally, thank you to all members of the Image Processing Group, who helped me with inspiration for my research and whose company kept me positive when my ideas were not working.

Abstract

The number of images captured every day increases as time goes on. Many of these images are taken outdoors, where weather effects can influence the quality of the captured image. The degradation caused by atmospheric effects can be problematic, both in terms of visibility for a human observer and reducing the efficacy of computer vision algorithms. Because of this, the task of restoring these images is essential. This thesis focuses on restoring images degraded by fog or rain. The current state-of-the-art methods in the field are all based on supervised learning. This is a problem because supervised learning is done on synthetic datasets, resulting in poor generalization to real data.

We propose a variational model for single-image dehazing based on a smooth approximation of the dark channel prior. We also formulate an unsupervised deep learning model for dehazing based on the same functional. We also propose a variational model for single-image rain removal. The proposed models do not require synthetic data, so they have better properties on real hazy and rainy images than the state-of-the-art models.

We compared the proposed methods to the state-of-the-art on real images degraded by weather, and our methods proved to be better in restoring the images. The comparison was made subjectively by visual inspection and objectively by comparing the improvement in object detection performance of restored images.

Keywords: image restoration, variational methods, image dehazing, image deraining

Prošireni sažetak

Varijacijske metode za obnavljanje slika degradiranih maglom i kišom

Razvoj tehnologije dozvoljava sve veću automatizaciju raznih procesa. Povećanje automatizacije istovremeno povećava važnost automatske analize i obrade slike. Problem koji se pojavljuje je da velika većina algoritama za analizu slike optimalno funkcionira samo na slikama s dobrom vidljivosti. Vremenske neprilike kao što su kiša i magla mogu smanjiti vidljivost scene na slici, a time smanjiti i efikasnost algoritama za obradu slike. Zbog toga je bitan razvoj algoritama za detekciju neprilika poput magle i kiše i algoritama za uklanjanje efekata tih vremenskih neprilika iz slika. Ovi algoritmi mogu biti bazirani na videozapisu, to jest na nizu slika, ili na zasebnim slikama. Fokus ovog istraživanja je na metodama koje se baziraju na jednoj slici bez pretpostavki o postojanju dodatnih informacija i drugih slika.

Prvi korak u razvijanju algoritama za popravljanje slika je razvijanje modela degradacije. U slučaju magle, degradacija nastaje zbog raspršivanja svjetlosti na mikroskopskim česticama suspendiranima u zraku između kamere i scene. Glavni parametri koji utječu na to koliki će biti utjecaj magle su gustoća čestica koje raspršuju svjetlo, to jest gustoća magle, i udaljenost koju svjetlo treba proći od objekta do kamere. Objekti koji se nalaze dalje od kamere bit će više zamućeni maglom u slici. Uz to, raspršeno svjetlo povećava svjetlinu cijele slike. To povećanje svjetline rezultat je uprosječivanja raspršivanja svjetla na velikom broju čestica. Problem uklanjanja efekta magle sa slike je problem procjene boje raspršenog svjetla i procjene razine zamućenosti svakog piksela. Kada procijenimo te dvije stvari, možemo ukloniti efekte magle. U većini modela u literaturi koristi se dodatna pretpostavka da se sve frekvencije podjednako raspršuju na česticama. Rezultat ove pretpostavke je da je jedna procjena efekta magle dovoljna za sva tri kanala slike. Iako pretpostavka nije savršeno fizikalno točna, dovoljno je dobra aproksimacija za većinu primjena.

U slučaju kiše, degradacija nastaje zbog raspršivanja svjetlosti na kapljicama kiše. Vizualno, degradaciju možemo opisati kao velik broj svijetlih paralelnih crta na slici. Crte nastaju zbog kretanja kapljice za vrijeme ekspozicije kamere, a paralelne su jer gotovo sve kapljice padaju u istom smjeru. Da bismo uklonili ovaj efekt kiše, potrebno je detektirati crte uzrokovane kišom u slici i ukloniti ih, ali istovremeno ne ukloniti rubove stvarnih objekata koji imaju istu orijentaciju kao kiša. Kako crte o kojima govorimo nastaju zbog kretanja kapi kiše za vrijeme ekspozicije, kamere sa izuzetno malim vremenom ekspozicije ne mogu reproducirati taj efekt. To čini skupljanje baze slika sa kišom dodatno kompliciranim budući da velik broj modernih kamera ima kratko vrijeme ekspozicije.

Zanimljivo je napraviti i usporedbu ova dva tipa degradacije u frekvencijskom prostoru. Većina tipova šuma kojima se bavi računalni vid primarno imaju efekt u visokim frekvencijama

slike. Neki od tih tipova šuma su impulsni šum, Gaussov šum, zamućenje zbog kretanja ili zamućenje zbog fokusa. U smjeru okomitom na smjer padanja kiše, kiša ima efekt gotovo identičan kao impulsni šum. U tom smjeru zato i kiša ima primarno efekte u visokim frekvencijama. U smjeru padanja kiše postoji jači efekti na niže frekvencije, a razina efekta je ovisna o duljini vremena ekspozicije. Magla, za razliku od toga, ima većinu svog efekta sadržanu u niskim frekvencijama slike. Taj efekt vidimo u povećanju svjetline cijele slike i smanjenju kontrasta i zasićenosti boja. To čini efekte magle različitima od većine tipova šuma, pa su i metode kojima se uklanjaju efekti magle različiti od metoda za druge tipove šuma.

Najuspješnije moderne metode u većini područja računalnog vida bazirane su na dubokom učenju. To je istina i za uklanjanje efekata degradacija magle i kiše. Problem s kojim se susrećumnoge od tih metoda je to što u fazi treniranja trebaju parove čistih i degradiranih scena, to jest potrebno je imati sasvim identičnu scenu s efektom magle odnosno kiše i bez tog efekta. Da bi zaobišli taj problem, mnogi pristupi bazirani na dubokom učenju koriste sintetičke skupove podataka. To su skupovi podataka u kojima su slike s degradacijom umjetno generirane koristeći matematički opis efekata magle ili kiše i stvarne slike bez efekata vremenskih neprilika. Takav način generiranja skupova podataka dozvoljava treniranje modela koristeći nadzirano učenje, a istovremeno čini i objektivno uspoređivanje modela jednostavnijim. Nedostatak ovakvog pristupa je da modeli time uče kako ukloniti maglu i kišu generiranu matematičkom formulom, a ne nužno efekte stvarnih vremenskih neprilika. To nas je motiviralo da razvijemo metode koje nisu bazirane na nadziranom učenju. Bavili smo se primarno varijacijskim metodama, to jest metodama baziranima na minimizaciji funkcionala dizajniranog tako da njegov minimizator bude slika s uklonjenim utjecajem vremenskih neprilika. Da bismo mogli pošteno usporediti metode koje smo razvili sa trenutno najuspješnijim metodama istražili smo i skupove podataka sa stvarnim degradacijama maglom ili kišom za koje je moguće odrediti kalitetu rekonstrukcije. Za maglu smo koristili skup podataka slika sa prometnica i iz urbanih sredina za koje su dostupne anotacije detekcija pješaka i automobila. To nam je omogućilo sve metode usporediti na zadatku predprocesiranja slike za detektor objekata koristeći slike pod utjecajem stvarne magle. Za kišu smo koristili skup slika rekonstruiranih iz kratkih videosnimaka statične scene iz kojih je moguće rekonstruirati kako scena izgleda bez kiše, a u kojima je moguće uzeti jednu sliku iz videa te na njoj testirati uklanjanje kiše.

Prije razvijanja metode za uklanjanje magle, napravili smo detaljnu matematičku analizu propagacije greške u uklanjanju magle. Koristeći model degradacije maglom izveli smo ovisnost greške u konačnoj rekonstrukciji slike bez utjecaja magle o grešci u procjeni parametara magle. Ukoliko podcjenimo udaljenost objekta od kamere, rezultat će biti da dio efekta magle neće biti uklonjen sa tog dijela slike. Ukoliko precijenimo tu udaljenost, rezultat će biti prezasićen dio slike i često će ta prezasićenost dovesti do u potpunosti crnog ili u potpunosti bijelog dijela slike. Takav gubitak informacija je značajno veći problem od blagog preostalog

efekta magle. Također smo utvrdili da se greška u procjeni dubine i gustoće magle najbolje mjeri multiplikativno, a greška u procjeni boje magle aditivno. Dodatno, zaključili smo da prevelika procjena dubine, čak i kada je gotovo sasvim ispravna, može pojačati efekte greške u procjeni boje magle, i da je zato potrebno ograničiti maksimalnu procjenu dubine u svim našim metodama. Efektivno ovime izbjegavamo rizik dijeljenja nulom. Neke od rezultata ove analize koristili smo u dizajnu modela za uklanjanje magle, a neki rezultati su za sada ostali samo teorijski uvidi u prirodu degradacije slike maglom i problema uklanjanje te degradacije.

Za uklanjanje magle razvili smo varijacijsku metodu baziranu na glatkoj aproksimaciji pretpostavke o vrijednosti minimalnog kanala. Na toj su pretpostavci bazirane neke od najuspješnijih klasičnih metoda, ali zbog matematičkih svojstava same pretpostavke nije ju lagano kombinirati s drugim metodama. Pretpostavka je bazirana na minimalnoj vrijednosti slike u okolini svakog piksela, a kako niti minimum niti lokalizacija nisu glatki operatori, nije niti čitava pretpostavka. Da bismo glatko aproksimirali ovu pretpostavku koristili smo konvoluciju sa karakterističnom funkcjiom kvadrata oko ishodišta koja je igrala ulogu lokalizacije i p-normu za aproksimaciju ekstrema. Razvijena aproksimacija o minimalnom kanalu je u obliku koji je jednostavnije kombinirati s drugim metodama. Čitav funkcional na kojem se bazira ova varijacijska metoda sastoji se od četiri sumanda. Prvi osigurava da rekonstruirana slika do neke razine zadovoljava model degradacije magle. Drugi je totalna varijacija dubine slike, čime se osigurava glatkoća procjenjene dubine na slici. Treći sumand čini funkcional pristranim da podcjenjuje dubinu slike radije nego ju precjenjuje, što smo zaključili da je bolje u analizi modela magle. Četvrti i zadnji član je glatka aproksimacija pretpostavke o minimalnom kanalu. Kao drugi primjer korištenja te glatke aproksimacije razvili smo i neuronsku mrežu treniranu nenadziranim učenjem koja se bazira na prilagođenoj formi istog funkcionala kao varijacijska metoda. Jedna velika prednost ove metode bazirane na nenadziranom učenju je ta što ne sadrži pretpostavku da je boja magle identična posvuda u slici, što dozvoljava funkciji da bolje funkcionira u situacijama sa kompleksnim osvjetljenjima zamagljene scene, kao što je na primjer noć sa nekoliko svjetiljki.

Za uklanjanje kiše razvili smo varijacijsku metodu baziranu na težinskoj usmjerenoj totalnoj varijaciji. Metoda se sastoji od tri koraka. Prvi korak je automatska procjena kuta pod kojim kiša pada. To postižemo time što prvo segmentiramo sliku morfološkim operacijama na komponente koje bi mogle biti kiša i zatim metodom principijalnih komponenti odredimo smjerove dobivenih povezanih komponenti. Smjer koji se najviše puta pojavljuje među komponentama je aproksimacija smjera pada kiše. Morfološke operacije kojima dobivamo te povezane komponente se baziraju na pretpostavkama da je kiša svjetlija od njene okoline u slici i da su tragovi kiše mnogo duži nego su široki. Drugi korak je određivanje težina koje ćemo primjeniti u samom funkcionalu. Cilj je svakom pikselu pridružiti težinu koja govori o tome koliko vjerojatno smatramo da je na tom piskelu vidljiv utjecaj kiše. Cilj tih težina je u očuvanju rubova

u slici koji su orijentirani isto kao smjer padanja kiše, ali nisu uzrokovani kišom. Veoma jednostavni načini procjenjivanja ovih težina pokazali su se uspješnima, na primjer uzimanje vrijednosti najmanjeg kanala i uzimanje vrijednosti saturacije piksela. Treći je korak minimizacija ovako dobivenog funkcionala. Funkcional se sastoji od tri sumanda. Prvi je usmjerena totalna varijacija rekonstruirane slike u smjeru pada kiše pomnožena sa procjenjenim težinama. Ovaj član osigurava da se u rekonstruiranoj slici ne nalazi kiša. Drugi član je 1-norma razlike rekonstruirane slike i originalne slike. Rezultat ovog člana je da su orginalna i rekonstruirana slika identične u velikom broju piksela, što odgovara lokalnosti efekta kiše na slici. Treći sumand je totalna varijacija procjenjene slike kiše u smjeru okomitom na smjer pada kiše. To penalizira postojanje rubova u kiši koji nisu u smjeru kiše, što rezultira boljim očuvanjem rubova u rekonstruiranoj slici. Hiperparametri koji kontroliraju težinu svakog od ovih sumanada također ovise o samoj procjenjenoj matrici težina, kako bi metoda bila prilagodljiva različitim razinama kiše. Samu minimizaciju radimo metodom augmentiranog Lagrangiana i metodom alternirajućih smjerova multiplikatora. U koraku gradijentnog spusta s obzirom na varijablu po kojoj je funkcional gladak koristimo iterativnu metodu rješavanja dobivenog linearnog sustava kako bismo ubrzali metodu. Velika prednost metode pred postojećim varijacijskim metodama je u tome što naša metoda ima bolja svojstva očuvanja rubova paralelnih s kišom koji nisu nastali kao efekt kiše.

Razvijene metode usporedili smo s najuspješnijim dostupnim metodama baziranima na nadziranom učenju. U usporedbi na sintetičkim skupovima podataka, metode trenirane nadziranim učenjem pokazale su se značajno bolje od naših predloženih metoda. To je bilo očekivano, jer su metode trenirane nadziranim učenjem na točno takvim skupovima podataka. U usporedbi na pravim podatcima, naše metode pokazale su se značajno bolje. Eksperimenti u kojima smo koristili metrike za koje nije nužna referentna slika su pokazali da metode bazirane na nadziranom učenju gotovo uopće ne utječu na prave slike degradirane vremenskim uvjetima ili ih čak malo pogoršavaju. Najuspješnija se pokazala metoda bazirana na minimizaciji funkcionala, a malo manje uspješna predložena metoda bazirana na nenadziranom učenju. Isti smo rezultat dobili i u eksperimentima gdje smo provjeravali koliko uklanjanje magle poboljšava detekciju unaprijed istreniranog detektora objekata. Za te eksperimente koristili smo slike sa prometnica slikane u magli i isprobavali promjenu u uspješnosti detekcije pješaka, atuomobila i drugih vozila. Za obje metode koje smo mi razvili razina detekcije je povećana, a metode bazirane na nadziranom učenju su ili manje povećavale ili čak smanjivale preciznost detektora. Ovakav eksperiment može biti od velike važnosti, pogotovo za primjene u primjerice autonomnim vozilima. Usporedili smo i složenost predloženih metoda sa metodama baziranima na nadziranom učenju. Naša predložena metoda bazirana na minimizaciji funkcionala vremenski je najzahtjevnija, jer je potrebno eksplicitno minimizirati funkcional za svaku zasebnu sliku, ali naša metoda bazirana na nenadziranom učenju pokazala se nekoliko

puta bržom od metoda baziranih na nadziranom učenju. Isto vrijedi i za usporedbu memorijske složenosti svih spomenutih metoda.

U eksperimentima za uklanjanje kiše ponovili smo eksperimente na skupovima podataka koji su umjetno generirani iz čistih slika. Zanimljiva usporedba je bila i provjeriti kako metoda trenirana na jednom skupu podataka generalizira na drugi skup, budući da kiša nije na svim skupovima identično generirana. Osim toga, u uklanjanju kiše napravili smo dodatni eksperiment u kojem smo koristili metode za uklanjanje kiše iz videa, te njihov rezultat koristili kao referentnu sliku za uklanjanje kiše iz jedne slike tog videa. U ovom se eksperimentu također naša metoda pokazala boljom od metoda baziranih na nadziranom učenju.

Zaključak ove disertacije je da pristup obnavljanju slike baziran na modelima može biti efikasniji nego pristup baziran samo na podatcima. To je pogotovo točno za primjere u kojima su podatci generirani, a ne stvarni, jer tada metode nadziranog učenja ne mogu naučiti ništa o stvarnoj degradaciji. Pristupi bazirani na modelu degradacije u kombinaciji sa nenadziranim učenjem su se pokazali pogotovo pogodni.

Ključne riječi: obnavljanje slika, varijacijske metode, uklanjanje magle, uklanjanje kiše

Contents

Chapter 1

Introduction

As cameras are becoming more and more common, the number of captured images increases daily. Many of these images are captured outdoors, for example, by people taking pictures with their phones or by security cameras filming streets. In many cases, the weather can negatively affect these outdoor images. This degradation of quality due to weather can reduce the visibility of an image for a human observer. It can also make image processing and computer vision algorithms less efficient on those images. This can decrease the precision of many important computer vision tasks, such as object detection, image segmentation, calculation of optical flow, and many others. As an example, in Figure 1.1 we can see the features of the SIFT algorithm [1] calculated on an image without rain and on an image to which rain was added. Note how many of the detected points of interest are actually rain streaks and not features of the scene itself. This can make it much more difficult to use these features. Because of this, the problem of inverting these weather-caused degradations and restoring a clear image is important. This image processing task is a part of the field of image restoration. In general, image restoration is a field of image processing that deals with restoring images that have been degraded according

Figure 1.1: SIFT features calculated on an image without rain (left) and on an image with rain (right). Note how many of the detected features are actually features of rain rather than the scene itself.

to a known model in advance. Mathematically, this can be written as

$$I = f(J; p) \tag{1.1}$$

Here I is the observed, degraded image, and J is the clear image, sometimes referred to as the ground truth. The function f, which depends on some parameters p, is a model of degradation. This model is generally derived from the physics of what caused the initial degradation and is problem specific. In this thesis, we will deal with problems of degradation caused by haze and rain, which have very different degradation models. While the form of the model is generally known ahead of time, the exact values of parameters p are often not available, and estimating them is a part of the image restoration task. Mathematically, the task is first to estimate $\hat{p}$ as an approximation of real p and then find a stable approximation of the inverse of f. Using these two things, we can recover an approximation of the clear image J.

1.1 Variational methods

Variational methods are a class of methods for image processing based on minimizing a specific functional. This functional is designed so that the input which minimizes it is the solution to a specific problem. Usually these functionals have multiple terms, commonly called *data term* and *regularization terms*. The data term describes the model of the degradation and ensures that the solution follows the degradation model. The regularization terms are additional functions that encode in the functional some assumptions we have about the solution. For example, if we take the problem of image denoising for additive noise, the model is

$$I = J + \varepsilon \tag{1.2}$$

For a regularizer, we will assume that the original image is smooth, or in other words, the gradient of the original image is relatively small. The functional we would want to minimize in this case would be

$$L(S) = \min_S ||I - S||_2^2 + \lambda ||\nabla S||_2^2 \tag{1.3}$$

The first term is the data term, and the second term is the regularizer. The S that minimizes this functional is a guess of J based on our assumptions. For a different example, if we are solving the problem of super-resolution, the assumption is that the low-resolution image I comes from a high-resolution image J. The model of degradation would be

$$I = D(J) \tag{1.4}$$

Here D is a downsampling operator. If we again use the smoothness of the image as a regularizer, the functional now becomes

$$F(S) = ||I - D(S)||_2^2 + \lambda ||\nabla S||_2^2 \tag{1.5}$$

The parameter λ that appears in both of these examples is called a hyperparameter and is used to determine how important a particular regularizer is. The larger the λ, the more influence it has on the minimizer of the functional.

1.1.1 Euler-Lagrange equations

There is still the important question of how to minimize the functionals. Assume that the functional in question is of the form

$$F = \int L(x, y, f(x,y), f_x(x,y), f_y(x,y)) \tag{1.6}$$

where x and y are spatial coordinates, f the function, usually image, with respect to which we are optimizing and f_x and f_y denote derivatives of f with respect to x and y. Then, the Euler-Lagrange formula for the gradient of this functional is given as

$$\frac{\partial L}{\partial f} - \frac{\partial}{\partial x}\left(\frac{\partial L}{\partial f_x}\right) - \frac{\partial}{\partial y}\left(\frac{\partial L}{\partial f_y}\right) \tag{1.7}$$

This is commonly used by setting the gradient equal to 0 and solving the resulting system of differential equations to find critical points explicitly and, from there, to find the minimums. When the resulting equations are too complex to solve explicitly, an alternative way is to use the gradient in an optimization scheme such as gradient descent and find the minimum that way. The important part of these equations is that it gives us a way to calculate the gradient even if the functional includes the derivatives of the function f. This is particularly useful for regularizations such as total variation, which depend on the gradient of f.

1.1.2 Augmented Lagrangian

The Euler-Lagrange equations can be a valuable tool for solving optimization problems. However, they have two significant drawbacks. The first one is that they require all of the functions in them to be differentiable. This is not too large of a problem, as in most cases we can approximate non-smooth functions with smooth ones, but it can complicate things sometimes. The second limitation is that they only work for unconstrained optimization. In cases of constrained optimization, we need different tools.

The most commonly used tool for constrained optimization is the Lagrangian. If the given

optimization problem is

$$\begin{aligned} &\min_x f(x) \\ &\textbf{such that } c_i(x) = 0 \quad \forall i \end{aligned} \tag{1.8}$$

The Lagrangian of that problem is given as

$$L(x, \lambda) = f(x) + \lambda c(x) \tag{1.9}$$

Finding the stationary points of this function L is a way of finding the minimum of the problem from equation (1.8). In practice, there are other methods to do this. One way is the penalty method. In the k-th step of the penalty method, we find the solution to the unconstrained problem of minimizing L_k given by

$$L_k = f(x) + \mu_k \sum_i c_i(x)^2 \tag{1.10}$$

With the additional assumption that μ_k go to ∞, this can give the solution to the problem. However, the increasing values of μ_k can cause numerical instability, and the convergence can be slow even when numerically stable. To solve this, we use the augmented Lagrangian. It adds an additional term to the penalty method that mimics the Lagrangian multipliers. The kth step of the augmented Lagrangian method is solving the problem

$$\min L_k(x) = f(x) + \mu_k \sum_i c^2(x) + \sum_i \lambda_i c_i(x) \tag{1.11}$$

After each iteration, the estimates of the Lagrange multipliers λ_i are updated

$$\lambda_i \leftarrow \lambda_i + \mu_k c_i(x_k) \tag{1.12}$$

The μ_k can also be updated as in the penalty method, but they do not have to go to ∞. This allows us to avoid numerical instability problems, and μ_k can even be left fixed.

In order to solve more complicated optimization problems, we will often use the alternating directions method of multipliers (ADMM) together with the augmented Lagrangian. This method essentially decouples the optimization process into two separate and simpler steps. Assume we have an optimization of the form

$$\min_x f(x) + g(x) \tag{1.13}$$

In order to simplify solving this, we can instead solve the problem

$$\min_{x,y} f(x) + g(y) \quad \textbf{such that } x = y \tag{1.14}$$

These two problems are obviously equivalent, but this is now written in the form of constrained optimization, so we can use the augmented Lagrangian:

$$L_k(x,y) = f(x) + g(y) + \mu||x-y||_2^2 + \lambda(x-y) \tag{1.15}$$

In each iteration, we can separately minimize the expression for x and y. In case f is smooth, the minimum is simply given by setting the gradient to 0, and so is given by the system

$$\nabla f(x) + 2\mu(x-y) + \lambda = 0 \tag{1.16}$$

However, this is solvable even in many cases when f or g are not smooth. The problem

$$\min_x f(x) + \mu||x-y||_2^2 \tag{1.17}$$

is a well-researched problem in convex optimization. It can be interpreted as the problem of minimizing $f(x)$ while still staying in the neighborhood of the initial value y. The solution to problem (1.17) is called the proximal operator and for many functions f there is even an explicit formula for this operator. For our purposes, the most important of these functions will be the absolute value function, or more generally, the 1-norm. The proximal operator of this function is the minimizer of expression:

$$\min_x ||x-y||_2^2 + \lambda||x||_1 \tag{1.18}$$

and is given by formula

$$prox_{l_1,\lambda}(x) = \text{sign}(x)\max(|x| - \lambda, 0) \tag{1.19}$$

This is also known as the soft thresholding operator.

1.2 Haze degradation

The effects of haze on capturing an image were first studied by Nayar and Narashima in [2]. The authors described a physical model based on light scattering off of particles suspended in air and how that produces a degraded image. The model they derived for the degradation of an image by haze is:

$$I = Jt + (1-t)A \tag{1.20}$$

Here I is the observed hazy image, J is the ground truth haze-free image, t is a transmission map that depends on the depth of the image and the density of the haze, and A is called airlight, and it describes the color of the haze. The transmission map t is given by the formula:

$$t(x) = e^{-\beta(x)d(x)} \tag{1.21}$$

Figure 1.2: An example of a hazy image generated from a clear image using equation (1.20).

Here β is a variable that describes the density of the haze, and d is the distance between a given point and the camera taking the image. This means that t can take values between 0 and 1. The value $t = 0$ would happen only when $\beta(x)d(x) = -\infty$, which would mean either effectively infinitely dense fog or a point that is infinitely distant from the camera. This would result in $I = A$ for those points, meaning that no information is known about this point. The value $t = 1$ would mean either $\beta = 0$ or $d = 0$, meaning that there is no haze or that the point is right at the camera. For that value we have $I = J$, meaning that the observed image is not degraded by haze at that point.

The first term of this model is the attenuation part. Not all of the light reflected from an object will reach the camera, or the observer in general. This fraction of light that will be scattered increases with both density of the particles causing the haze and with the distance light has to travel. The second term of the model is the influence of the airlight. The airlight is an attempt to take into account the light scattered off of its initial trajectory. All of this diffused light averages out to give fog the light grayish color it has.

An important assumption in this model is that the transmission map t is the same for all three image channels R, G, and B. This is an approximation that is not fully physically true since light of different wavelengths scatters differently, but the approximation is good enough for purposes of image dehazing.

While this model does a good job of describing the effects of haze on the saturation and brightness of an image, it does have its shortcomings. Most notably, when we look at a direct source of light through a haze, we can see a ring of light diffused off of the source. This effect is not captured by this relatively simple model. However, even without these diffusive properties, this model can generate good, realistic-looking, hazy images from images with known depth. Figure 1.2 shows an example of such a generated synthetic hazy image. It is generated from an *RGBd* image, in other words, an image for which depth is known at each pixel, by manually choosing values of β and A and the using equation (1.20). For purposes of generating images like this, we will usually assume that both β and A are constant across the entire image.

1.3 Rain degradation

The most common way for rain to degrade an image is through the appearance of rain streaks. Rain streaks are light gray lines that appear in an image captured during rain. These lines appear because light refracts differently when it hits a raindrop, and raindrops move during the exposure time. The movement of the raindrop leaves the line in the image. The most common way this degradation is modeled is:

$$I = J + R \tag{1.22}$$

Once again, I is the observed image, J is the ground truth image, and R is the effect of the rain. Because R is generated by raindrops falling during the exposure time, we can make some assumptions about its form. One assumption is that R has value 0 in most pixels, as rain only affects some pixels of the image. Another assumption is that R contains mostly lines and that lines are all directed in the same general direction. The direction assumption holds because most raindrops will fall in a similar direction. This direction is mostly straight down or slightly to the side if there is a strong wind blowing.

An additional assumption that is often made is that the pixels affected by rain are brighter than those around them. This is not mapped directly to R being white because a rain streak in front of a blue sky will result in a white rain streak in the image, but the rain layer R will be closer to red than white.

1.4 Spectral analysis of atmospheric degradation

It can often be useful to perform spectral analysis of noise added to an image using the Fourier transform. Most commonly occurring noises, such as impulse noise and Gaussian noise, are primarily high-frequency noises and are treated differently from noises that also damage low-frequency parts of an image. Since high frequencies contain the details, while low frequencies contain the main shape of the image, this kind of analysis can also help us determine what type of change to expect from a degradation. If we apply the Fourier transform to equation (1.20) we get:

$$\hat{I} = \hat{J} * \hat{t} + (\delta - \hat{t}) * \hat{A} \tag{1.23}$$

Since A is close to constant, $\hat{A}$ is very close to a δ distribution. The transmission map t is related to the depth map and has the Fourier transform similar to that of a natural image but without small details. This means that $\hat{t}$ is also focused in lower frequencies, but less so than $\hat{A}$. The term $\delta - \hat{t}$ is still mostly described by low frequencies, and so the convolution $(\delta - \hat{t}) * \hat{A}$ will be mostly contained in low frequencies, but slightly blurred by the convolution. If we assume that A is constant, the convolution becomes simply multiplication with that constant. As a result, the

haze will affect the image I in low frequencies, and the term $\hat{J} * \hat{t}$ will impact high frequencies, making the details less pronounced. From this, we can conclude that haze as a degradation affects the entire Fourier space and is even focused in the lower frequencies rather than being mostly contained in the high frequencies.

On the other hand, rain is modeled by a linear equation 1.22. The Fourier transform is then just

$$\hat{I} = \hat{J} + \hat{R} \tag{1.24}$$

So the spectral analysis of rain degradation is reduced to just the spectral analysis of the term R. As discussed previously, R is most commonly modeled as motion blur of a raindrop, or in other words, a convolution of impulse noise with a directed kernel. This also means that in the direction perpendicular to the direction of rainfall, the effects of rain are fully equivalent to the effects of impulse noise. While in the direction of rainfall, rain degradation affects the lower frequencies a bit, it is still mainly contained in the high frequencies, especially so for the direction perpendicular to the direction of rainfall.

In conclusion, while the effects of rain are slightly more structured, they are still relatively standard high-frequency noise. However, the effects of haze degradation primarily affect the low frequencies, making it one of the few types of degradation that works that way. For this reason, the problem of dehazing is more interesting to explore, as none of the standard methods of denoising can be easily adjusted to it.

Chapter 2

Prior Work

For both rain removal and image dehazing, there is a general division into two separate problems. The first problem is the one we are solving in this thesis, the problem of single image dehazing and single image rain removal. In this problem, the task is to recover a clear image from a single degraded RGB image. In general, no additional information about the image, such as depth, is available. This is a realistic setting in many scenarios in the real world. There is also a second similar problem, the problem of video dehazing or rain removal from a video. In these problems, we are given a stream of images, and the goal is to remove the degradation from all of them combined. Obviously, this problem offers more information than the single image problem. This additional information is particularly useful for the rain removal problem due to the high-frequency nature of rain streaks. For haze, the problem of video dehazing is very closely connected to single image dehazing, as there is very little additional information available from previous frames. In the rest of the thesis, we will deal solely with the single image approaches.

For image restoration, there are, broadly speaking, three different approaches. The first one is the classical methods. These methods are usually based on hand-crafted priors and are given by a specific algorithm. They are called classical because they were developed in the classic ways of computer vision rather than being purely data-driven. The second approach are variational methods. They are closely related to classical methods but are generally phrased as a problem of minimization of a functional, as discussed in the previous section. The last approach is the data-driven one, most commonly based on deep learning. The majority of deep learning methods for image dehazing or image rain removal are based on supervised learning. This poses a problem, as they depend on many pairs of clean and degraded images for training, which are often unavailable. This will be discussed further in section 3.

2.1 Single image dehazing

2.1.1 Classical methods

The most widely used classical method is based on the **dark channel prior** [3]. This prior roughly states *near every pixel there exists a pixel such that a least one of its channels has the value close to* 0. This assumption, if written mathematically, is

$$\forall x \min_{y \in \Omega_x} J(y) \approx 0 \tag{2.1}$$

In this equation, Ω_x denotes a patch around the point x. Haze, with its effect of increasing brightness, causes images to lose this property. We can use this to estimate a rough approximation of the transmission map t. First, we assume that the transmission is constant within the patch Ω. We can then estimate $t(x)$ as

$$t(x) = 1 - \frac{\min_{y \in \Omega_x} I(y)}{A} \tag{2.2}$$

This is the result of using the assumption from equation (2.1) in the equation (1.20). Using this, we can get a relatively good estimate of t at some points, namely those where the assumption of transmission map being constant within the patch Ω is a good approximation, but in other parts, there will be large errors. The transmission map estimated this way will be blocky, and the reconstruction done using it will cause halo effects around objects in the foreground due to the lower value of t being correct within those patches. In order to solve this, the authors used guided image filtering [4]. This is a method for smoothing out an image while respecting the edges of a reference image. This allows us to create a refined estimate of t and use it for the final reconstruction. This results in heavily reduced halo effects. However, it can still have problems with somewhat dense branches, as the initial estimate of t will not have any small values to propagate using guided image filtering.

Guided image filtering works by taking an image S_1 that needs to be smoothed and an image S_2 that is used as a reference image. In the case of dehazing, the rough estimate of t is the image that needs smoothing, and the hazy image I is the reference image. The filter works patchwise. We will model the filtered version of S_1 as a linear function of S_2 in every patch, as this ensures the preservation of edges from S_2 in S_1. In other words $S_1 = a_k S_2 + b_k$ Inside the k-th patch. The coefficients a_k and b_k are determined by minimizing the functional

$$E(a_k, b_k) = \sum((a_k S_1 + b_k - S_2)^2 + \varepsilon a_k^2) \tag{2.3}$$

The last thing we need to define the DCP algorithm fully is how to estimate A. This is done by taking the 0.1% pixels with the highest value in their minimal channel. These are the pixels

assumed to be the most influenced by haze. The value of A is then estimated as the mean value of these pixels.

This approach has been highly successful in dehazing, and many modifications of it have been made. This includes speeded-up versions of it [5], combining the dark channel prior with the Wiener filter [6] and combining the dark channel prior with a prior set on the brightest channel [7].

A second very successful classical method is based on the haze lines called **non-local image dehazing** [8]. This is based on the observation of color clustering in non-hazy images. When no haze is present, most objects in the scene have relatively few colors. This causes the distribution of pixels in the *RGB* space not to be uniform but grouped into a small number of clusters. If we view the haze degradation model (1.20) in the *RGB* space, the pixel is moved from its original value $J(x)$ towards the value of A along a line. The transmission $t(x)$ controls how far the pixel is moved along the line. This causes the clusters of colors in the natural image to elongate into lines. The idea behind this method is to detect these lines and push pixels back down them in the *RGB* space.

In order to estimate A, we will use Hugh transform in spherical coordinates. Spherical coordinates are used because the prior is that in the hazy image pixels all lie on lines that intersect in A. In order to estimate this, each pixel $I(x)$ will vote for every potential airlight A such that the distance from the line defined by θ and ϕ centered at A is close enough to the pixel. The distance can be calculated using the formula

$$||(A - I(x)) \times (\cos\theta, \sin\phi)|| \tag{2.4}$$

The value of A with the most votes is then chosen as the estimated airlight.

Now that A is estimated, we define

$$I_A(x) = I(x) - A \tag{2.5}$$

In terms of *RGB* space, this translates the A to the origin of the coordinate system. By using this in equation (1.20), it now becomes

$$I_A(x) = t(x)(J(x) - A) \tag{2.6}$$

If we again move to the spherical coordinates centered around A, $I_A(x) = [r(x), \theta(x), \phi(x)]$, the problem is reduced by one dimension. The lines of pixels we are trying to detect are fully defined by just θ and ϕ. Using a Hugh transform-like process, like in determining the value of A, each pixel votes for which line it lies on, and the lines with a number of votes above a certain threshold are assumed to be the true haze lines. Using these haze lines, a rough transmission

map is first estimated by assuming that the pixel furthest from the origin is fully haze-free. After that, the final transmission is again created by filtering the initial transmission while respecting the edges of the original image.

As the method is good at removing haze, there have also been many modifications, such as combining it with wavelets [9] and adapting the method for use in restoring underwater images [10].

An another prior based approach is called **color attenuation prior** [11]. This approach could also be considered a learning-based approach. This prior is based on the observation that haze increases the brightness of an image and reduces saturation. The assumption is then made that the scene depth can be approximated using a linear model of brightness V and saturation S from the HSV color space. Once the initial depth is recovered, it is filtered using the guided image filter. After that, A is estimated to equal the color of the deepest pixels in the scene. This works well for outdoor images with some pixels very far from the camera, but indoor images can cause problems for this method. The final reconstruction is done using equation (1.20). A big advantage of this model is its speed and the fact that the model is linear. Because of this and the linearity of the gradient, edges in the estimated transmission must always come from edges in the real image.

2.1.2 Variational methods

The **Enhanced variational image dehazing** (EVID) is a variational approach to single image dehazing based on maximizing contrast. First, we modify the gray world assumption to account for the effects of haze. The gray world assumption is a prior from computational color constancy that says that the average of almost any image tends to be gray. If we apply this to equation (1.20) we get, thanks to the linearity of the mean

$$\text{mean}(I) = \text{mean}(Jt) + \text{mean}((1-t)A) \tag{2.7}$$

We now need to make two assumptions. The first assumption is that the color of a pixel on a haze-free image is independent of the depth of that pixel. In other words, the assumption is that t and J are independent variables. This allows us to write

$$\text{mean}(I) = \text{mean}(J)\text{mean}(t) + \text{mean}((1-t)A) \tag{2.8}$$

If we also assume that the depth is uniformly distributed across the image, then $\text{mean}(t) \approx \frac{1}{2}$. Taken all together

$$\frac{\text{mean}(J)}{2} \approx \text{mean }(I) - \frac{\text{mean}(A)}{2} \tag{2.9}$$

Using one of the previously described methods to estimate A allows us to estimate the mean of J from the observed image I.

$$\mu_J := 2 \cdot \text{mean}(I) - A \tag{2.10}$$

Now we can write the functional that defines EVID as

$$E(S) = \frac{\alpha}{2} \sum_x (S(x) - \mu_J) + \beta \sum_x (S(x) - I(x)) - \frac{\gamma}{2} \sum_{x,y} w(x,y)|S(x) - S(y)| \tag{2.11}$$

The first term ensures the solution has a mean similar to J, and the second term that it is still similar to the input image. In the last term, the function w is a weight function that is larger if x and y are closer in the image and falls to 0 when they are far away from each other. This third term is called the contrast enhancement term. The gradient of the contrast enhancement term at point x is

$$\frac{\sum_y w(x,y)|S(x) - S(y)|}{\sum_y |S(x) - S(y)|} \tag{2.12}$$

The minimizer of the functional (2.11) is the contrast-enhanced, dehazed version of the original image.

A further extension on EVID is the **Fusion based variational image dehazing** (FVID). In FVID, we first define an extension to the EVID energy from equation 2.11 by penalizing overly bright pixels

$$E_{FVID}(S) = \frac{\alpha}{2} \sum_x (S(x) - \mu_J) + \beta \sum_x (S(x) - I(x)) - \frac{\gamma}{2} \sum_{x,y} w(x,y)|S(x) - S(y)| + \tau \sum_x S(x) \tag{2.13}$$

However, rather than simply minimizing this functional, we will use all the different steps from the gradient descent. The starting point of the gradient descent is $S_0 = I$, and at each iteration, we have

$$S_{k+1} = S_k - \nabla E_{FVID}(S_k) \tag{2.14}$$

We then compute the differences in saturation after every step. The idea here is that closer regions have more saturated pixels, and the change in their values can be driven rapidly by inter-channel contrast maximization. Distant regions are closer to gray, so that part of the gradient will be smaller for them. So we define

$$D_k = \text{Sat}(S_{k+1}) - \text{Sat}(S_k) \tag{2.15}$$

Where

$$\text{Sat}(S) = \frac{\max_c S - \min_c S}{\max_c S} \tag{2.16}$$

This map correlates with the depth map. We then use these depth maps for a guided fusion of

the iterates into a dehazed image by using a weighted sum

$$J = \sum_k D_k S_k \tag{2.17}$$

This fusion-based approach ensures that the clear image does not have oversaturated pixels, especially the ones that were not initially affected by the haze.

Variational Single Image Dehazing for Enhanced Visualization [12] is also a variational method. It acts on the YUV colorspace defined as

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.436 \\ 0.615 & -0.515 & -0.100 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{2.18}$$

The dehazing is only done in the Y component, as it is the main one that reflects luminance. The U and V channels mostly deal with color. This immediately reduces the problem from 3 dimensions to 1. Also, in order to make the problem simpler, we apply logarithm on equation (1.20) in order to switch from multiplication to addition:

$$\log(A - Y_I) = \log t + \log(A - Y_J) \tag{2.19}$$

If we denote $f = -\log(A - Y_J)$, $g = -\log(A - Y_I)$ and $d = \log t$ the problem becomes

$$d + g = f \tag{2.20}$$

Now the functional used in this method can be written down

$$\min_{d,g} \frac{1}{2}||d + g - f||_2^2 + \alpha R_1(d) + \beta R_2(g) \tag{2.21}$$

The terms R_1 and R_2 are regularization terms. Term R_1 is the regularization of the transmission map, and so it is defined as total variation, the most common way to regularize the transmission map

$$R_1(d) = ||\nabla d||_1 \tag{2.22}$$

The term R_2 is the regularization of the logarithm of difference between clear image and A. This regularization is used to ensure that edges in J must correspond to edges in I. This is done by modeling the gradients as linearly dependant on one another

$$\nabla f(y) = a(x)\nabla g(y) \quad \forall y \in \Omega_x \tag{2.23}$$

Here Ω_x is a small patch around x, and the parameter a is assumed to be close to constant within every patch. Using this, we defined the regularizers R_2 as

$$R_2(g,a) = \frac{1}{2}\int\int_{\Omega_x} ||\nabla f(y) - a(x)\nabla g(y)||_2^2 dydx + \int |a(x)|^2 dx \tag{2.24}$$

This regularization term penalizes a ever being too large and penalizes f and g not having a close to linear relationship. However, an important thing to note is that a is allowed to be close to 0, so ∇g is allowed to be large where ∇f is small, but ∇g cannot be small where ∇f is large. The minimization of this functional can be efficiently done using the alternating directions method of multipliers and augmented Lagrangian, using the fast Fourier transform to solve the large linear systems resulting from the 2-norms and proximal operators for the problems that include the 1-norms.

The deep image prior [13] is a special kind of variational method for denoising. It is a variational method, but it also uses a convolutional neural network. It is based on the assumption that, due to the nature of convolutions, convolutional neural networks generate noise-free natural images more easily than noisy images. Formally, the method is based on minimizing

$$\underset{\theta}{\operatorname{argmin}} ||f_\theta(z) - I||_2^2 \tag{2.25}$$

Here f_θ denotes a neural network with parameters θ, and I is the observed noisy image. The network is trained to map a fixed noise z into I using gradient descent and, using early stopping mechanisms, the optimization can be stopped in such a moment that $f_\theta(z)$ is a good approximation of a denoised I. This assumption that neural networks generate natural images more easily than noisy images.

The **double DIP** [14] is a method for single image dehazing using multiple deep image priors simultaneously. Three networks are trained simultaneously, one generating the clear image, one generating the transmission t, and one generating the airlight A. The loss function used is of the form

$$\begin{aligned} L(\theta_1,\theta_2,\theta_3) = &||I - f_{\theta_1}(z)f_{\theta_2}(z) - (1 - f_{\theta_2}(z))f_{\theta_3}(z)||_2^2 \\ &+ ||\text{blur}(f_{\theta_3}(z)) - f_{\theta_3}(z)|| + ||f_{\theta_3}(z) - A_0||_2^2 \end{aligned} \tag{2.26}$$

Here f_{θ_1} is the network generating the clear image, f_{θ_2} is the network generating the transmission map t and f_{θ_3} is the network generating the airlight A. The first term is the data term and ensures that the generated image, transmission map, and airlight adhere to the haze model at least somewhat. The term $\text{blur}(A)$ is defined as a convolution of A with a 5×5 kernel of constant values $\frac{1}{25}$. It promotes the smoothness of the airlight. The A_0 is an initial estimate of the airlight, usually made using the dark channel prior method, and the entire third term is made

to ensure that the airlight is everywhere similar to the initial estimate. The regularization on A exists because the ground truth of A is much smoother than ground truths of J and t, so the prior of being generated by a convolutional neural network is not sufficient.

Even though there is no additional regularization on J or t, the method gives good results. Also, in this variant, unlike in the single deep image prior, there is no need for early stopping mechanisms, which simplifies the optimization procedure a lot. The optimization itself is done using a standard ADAM optimizer.

2.1.3 Deep learning

Since most deep learning-based approaches use supervised learning, there are two possibilities used for dehazing. One is to learn the mapping $I \to J$ directly using the ground truth and have the output of the network be the clear image. The other approach is to use deep learning to estimate the parameters t and A and then use them to estimate J. Since the models are trained on datasets generated using equation (1.20), the parameter estimation approach gives better results in most cases.

Dehazenet [15] is an example of a convolutional neural network for single image dehazing. It consists of several convolutional and maxpool layers resulting in a small overall number of parameters. The network is trained by choosing 16×16 patches of haze-free images and assuming that the content in them is independent of the depth. Then, the authors assume that the transmission t is constant within the patch and generate a hazy variant of the image. These patch pairs are then used to train the network using mean square error as the loss function. This was one of the first neural networks used for single image dehazing. It was the state-of-the-art on datasets available at the time but has since been surpassed.

The **Dehazeformer** [16] is a transformer for single image dehazing. The transformer architecture has seen great success recently, first in natural language processing and later in computer vision. The Dehazeformer estimates parameters t and A in order to estimate the reconstructed image J. Just like all transformers, it uses self-attention, but it does modify parts to improve efficiency for dehazing specifically. One big difference is removing the LayerNorm layer, as it reduces the precision of reconstruction. For the non-linear activation function, a custom soft approximation of ReLU is used

$$\text{SoftReLU}(x) = \frac{x + \sqrt{x^2 + \alpha^2} - \alpha}{2} \tag{2.27}$$

Setting $\alpha = 0$ this function becomes the ReLU function. In Dehazeformer the value $\alpha = 0.1$ is used. This is the current state-of-the-art result when it comes to most synthetic datasets used to test dehazing methods. It is trained using supervised learning.

The **Feature Fusion Attention Network** (FFANet) [17] is a convolutional neural network

for single image dehazing. It uses several convolutional blocks and both pixel-level attention and channel attention mechanisms. The channel attention is calculated as

$$CA_c = \sigma(Conv(\text{ReLU}(Conv(g_c)))) \tag{2.28}$$

Here c denotes the channels, $Conv$ is a convolutional layer, σ is the sigmoid non-linearity, and g_c is the global average pool of each layer. The output of this attention is multiplied elementwise with the input. The pixel attention is calculated as

$$PA = (Conv(\text{ReLU}(Conv(F^*)))) \tag{2.29}$$

The main difference between pixel and channel attention is that channel attention averages over space before convolutions, and pixel attention uses convolution to average over channels. The FFANet uses these mechanisms to directly estimate the dehazed image from a hazy one. The loss function used is the 1-norm of the reconstruction error, so the method has to be trained using supervised learning.

The **DehazeGAN** [18] is a slightly different approach to dehazing using deep learning in that it uses generative adversarial networks [19] (GANs). GANs are based on simultaneously optimizing two networks, a generator and a discriminator. One learns to generate the target image, the generator, and one learns to differentiate between real images from the target distribution and generated ones, the discriminator. This image-generation method has been very successful in many areas of computer vision. The CycleGAN [20] is a generalization of the GAN idea. Suppose we assume that there are two sets of images A and B. In that case, we can train a generator G_1 that maps images from A into images from B and a discriminator D_1 that differentiates real and generated images from the distribution of dataset B. We also train a generator G_2 that maps images from B to A and a discriminator D_2 that differentiates images in A. In order to link them all together, we also add the cyclic consistency term in the loss function

$$L_{\text{cyclic}} = ||I - G_2(G_1(I))||_2^2 \tag{2.30}$$

This term guarantees that when an image is mapped from A to B and then back to A, that it will be close to the same image. The DehazeGAN is a natural application of the CyclicGAN to dehazing, where A will be clear images and B will be hazy images. DehazeGAN also uses a perceptual loss using the hidden layers of the VGG [21] as an additional regularizer. The networks are trained on hazy and haze-free image pairs, with the generators learning to generate haze and remove it in the same way as using equation (1.20), but without explicit access to parameters t and A.

All-in-One Dehaze Network [22] (AoDNet) is a convolutional network for single image

dehazing that is also based on estimating the parameters of model (1.20), but it rephrases the parameters used slightly. The first step is to rewrite the model to a different form

$$I(x) = K(x)J(x) - K(x) + b \tag{2.31}$$

where

$$K(x) = \frac{\frac{1}{t(x)}(I(x) - A) + (A - b)}{I(x) - 1} \tag{2.32}$$

The network uses several convolutional blocks to estimate K and then uses the estimated K to reconstruct J. The model is also trained using supervised learning, with mean square error as the loss function.

2.2 Single image rain removal

Just like with single image dehazing, there are many ways to approach single image rain removal. There are many classical methods based on priors that assume the structure of the rain R. The variational methods are rarer than in dehazing because of the nature of the degradation, but they are also used. An approach that is not used in dehazing but is used in rain removal is dictionary learning, most commonly in the form of morphological component analysis. And, as in most areas of computer vision, deep learning has proven to be successful in the area of rain removal as well.

2.2.1 Classical methods

Classical methods are, again, the ones defined by a specific algorithm based on hand-crafted priors rather than being data-driven learning approaches. One of the most successful ones is **Rain Removal By Image Quasi-Sparsity Priors** [23]. It combines multiple priors about the form of rain layer R to identify pixels that potentially contain rain and then uses a sparsity prior to reconstruct the original image. The priors used to identify rain streaks are:

- rain streaks are usually much longer than they are wide
- the direction of all rain streaks in the scene is similar
- the color of a rain streak is usually close to white
- rain streaks are usually brighter than their surroundings

These priors are combined as follows. First, the value of every pixel is compared to the means of the five patches around it, one that it is the center of and four that it is one of the corners of. Then, for every pixel larger than all five means, the pixel at the same spot in a rain detection binary map is set to 1, and for others, it is set to 0. After this, the connected components of this rain detection map are identified. Each connected component is a potential rain streak.

However, the map also contains many misidentified bright objects. In order to filter those out, we will use the other priors. The first step is to calculate the PCA of each of the connected components. The ratio of principal values is called the aspect ratio, and if it is larger than a set threshold, the component is discarded as non-rain, as it does not have a fitting ratio of length and width. The second use of PCA is the first principal component. The first principal component is the principal direction in which that connected component is oriented. Since we assume that most rain streaks will be oriented in a similar direction, the connected components with orientation significantly different from the others are discarded as being non-rain. Finally, to filter out the remaining non-rain components, we will use the assumption on the color of the rain streaks. We denote with $\overline{R}$, $\overline{G}$, and $\overline{B}$ the mean of each channel in a single connected component. Let $\varphi = \frac{\overline{R}+\overline{G}+\overline{B}}{3}$. Now we can define values u and v as

$$\begin{aligned} u &= \frac{2\varphi - \overline{G} - \overline{B}}{\varphi} \\ v &= \max\left\{\frac{\varphi - \overline{G}}{\varphi}, \frac{\varphi - \overline{B}}{\varphi}\right\} \end{aligned} \tag{2.33}$$

Components with mostly neutral colors will cluster around $(0,0)$ in the (u,v) space. Those that are far from the origin are discarded as non-rain components. This gives a relatively good binary map classifying each pixel as either rain or non-rain. The deraining is now finished by using the rain detection map as a guide and minimizing a functional based on the first and second derivatives of the image.

Another prior that is sometimes used is the low-rank prior. In these methods, for instance, [24] and [25], the image patches are viewed as matrices. The prior states that the rain layer R will have a much lower rank when decomposed into patches than a natural image. This is justified by assuming that most rain patches will have values close to 0, while the non-zero values will be grouped into lines, resulting in low rank matrices. This is then combined with total variation denoising of the original image to decompose I into J and R.

2.2.2 Variational methods

The most successful variational method for single image rain removal is the **unidirectional global sparse method** (UGSM) [26]. It is based on minimizing the functional

$$\min_{R \leq S} \lambda_1 ||\nabla(S-R)||_1 + \lambda_2 ||R||_1 + ||\nabla_\perp R||_1 \tag{2.34}$$

The first term promotes solutions that are smooth in the horizontal direction, the second term ensures the sparsity, and thus localization, of the reconstructed rain layer, and the third term ensures that rain does not contain edges in the vertical direction. If needed, the image is rotated

to ensure the rainfall is directly in the vertical direction. The minimization of the functional is done using the augmented Lagrangian and the alternating directions method of multipliers. The method gives good results in terms of rain removal but has a significant drawback in that it can not differentiate between rain and general edges in the same direction.

Another variational method is an **L0-regularized global anisotropic gradient prior** for single image deraining [27]. it is based on a similar idea of the rain streaks inducing primarily gradient in a single direction. However, in order to maximize the sparsity of the gradient of the solution, this method uses a l_0 regularization. It is based on minimizing the functional

$$\min_{R\geq 0} \lambda_1||\nabla_x(I-R)||_0 + \lambda_2||\nabla_y(I-R)||_0 + \lambda_3||R||_0 + \lambda_4||\nabla_x R||_0 + \lambda_5||\nabla_y R||_0 \tag{2.35}$$

As in equation (1.20) I is the observed image, and R is the rain streak part of the image. The functional contains the l_0 pseudonorm and so cannot be regularized using standard tools of convex optimization such as augmented Lagrangian. However, similar ideas are used here. First, the functional is rephrased with new additional variables.

$$\begin{aligned} &\min \lambda_1||d_1||_0 + \lambda_2||d_2||_0 + \lambda_3||d_3||_0 + \lambda_4||d_4||_0 + \lambda_5||d_5||_0 \\ &\textbf{such that } d_1 = \nabla_x(I-R) \quad d_2 = \nabla_y(I-R) \quad d_3 = R \quad d_4 = \nabla_x R \quad d_5 = \nabla_y R \end{aligned} \tag{2.36}$$

This is minimized iteratively. The step for updating R is the same as in the augmented Lagrangian method, simply the 2-norms of all of the conditions

$$\begin{aligned} R^{k+1} = \operatorname*{argmin}_R &||\nabla_x(I-R^k) - d_1||_2 + ||\nabla_y(I-R^k) - d_2||_2 + \\ &||R^k - d_3||_2 + ||\nabla_x R^k - d_4||_2 + ||\nabla_y R^k - d_5||_2 \end{aligned} \tag{2.37}$$

Since this is smooth, and the gradient is linear, it can be solved explicitly. Due to the form of the gradient, the linear system can be solved using the fast Fourier transform. For updates of d_i, an equivalent of the proximal operator for the l_0 norm is used. The minimizer of the functional

$$\min_x ||x-y||_2^2 + \lambda||x||_0 \tag{2.38}$$

It is defined as

$$prox_{l_0,\lambda} = \begin{cases} y & y^2 \geq \lambda \\ 0 & \text{otherwise} \end{cases} \tag{2.39}$$

From this, the updates for d_1 to d_5 are defined as

$$
\begin{aligned}
d_1^{k+1} &= \begin{cases} 0 & \nabla_x (I - R^{k+1}) \leq \sqrt{2\lambda_1} \\ \nabla_x (I - R^{k+1}) & \text{otherwise} \end{cases} \\
d_2^{k+1} &= \begin{cases} 0 & \nabla_y (I - R^{k+1}) \leq \sqrt{2\lambda_2} \\ \nabla_x (I - R^{k+1}) & \text{otherwise} \end{cases} \\
d_3^{k+1} &= \begin{cases} 0 & R^{k+1} \leq \sqrt{2\lambda_3} \\ R^{k+1} & \text{otherwise} \end{cases} \\
d_4^{k+1} &= \begin{cases} 0 & \nabla_x R^{k+1} \leq \sqrt{2\lambda_4} \\ \nabla_x R^{k+1} & \text{otherwise} \end{cases} \\
d_5^{k+1} &= \begin{cases} 0 & \nabla_y R^{k+1} \leq \sqrt{2\lambda_5} \\ \nabla_y R^{k+1} & \text{otherwise} \end{cases}
\end{aligned}
\tag{2.40}
$$

The minimizer of this functional is the rain component R, and the clear image is $I - R$.

2.2.3 Dictionary learning

Dictionary learning is a denoising technique based on sparse representation. The idea behind it is to take a large training set of vectors I_i and find an overcomplete frame D, called a dictionary, such that the vectors I_i can be sparsely represented using vectors from D. Finding the sparse representation for a fixed dictionary is done using methods such as basis pursuit or matching pursuit. In our applications, the vectors I_i are images which are very high dimensional, with the dimensionality equal to the number of pixels times the number of channels. However, due to the highly structured nature of natural images, they can usually be represented using much fewer vectors.

Dictionary learning is used to separate the additive noise R from the ground truth image J. This is most commonly done using morphological component analysis (MCA). MCA is based on the idea of separating multiple images that are added together based on their morphological components using dictionaries describing each of them. If we assume that some image I is composed of sub-images I_1 to I_n, in other words

$$
I = I_1 + I_2 + ... + I_n \tag{2.41}
$$

the goal is to recover I_1 to I_n. To do that using MCA we learn dictionaries D_1 to D_n such that D_i describes I_i sparsely. Then, using a pursuit algorithm and the set $[D_1, D_2, ..., D_n]$, we can reconstruct I and separate it into different sub-images.

In [28], the authors propose, and later extend in [29], a way to use MCA for single image rain removal. First, the image is preprocessed to separate the low-frequency and high-frequency components. The low-frequency components should not contain any rain and so should not be used in the denoising process. This ensures that as much of the non-rain components as possible are preserved. Then, a dictionary D_{rain} is learned from examples of patches of rain. This dictionary is constructed using the kSVD method introduced in [30]. The result is a dictionary that can sparsely represent patches of rain. For describing the non-rain component, the authors chose curvelets, and so use dictionary D_{curvelet}. These two dictionaries are combined together into dictionary $D_{HF} = [D_{\text{curvelet}}|D_{\text{rain}}]$. After this, the problem of deraining is the problem of minimizing

$$E(I_{HF}, \theta_{HF}) = \sum \left(||b_{HF} - D_{HF}\theta_{HF}||_2^2 + \lambda ||\theta_{HF}||_1 \right) \tag{2.42}$$

Here I_{HF} represents the high-frequency parts of the image being derained. θ_{HF} are the coefficients in the dictionary D_{HF}. b_{HF} are the patches of the image I_{HF}, as the dictionaries are used to describe the separate patches rather than the entire image. The 1-norm is used to promote sparsity in the coefficients used. Once the optimal coefficients are found, the high-frequency component of the derained image is reconstructed using the coefficients in D_{HF} that correspond to the dictionary D_{curvelet}, while not using the ones that correspond to D_{rain}.

In [31], the authors propose a method that removes rain streaks from a single image using a hierarchical approach combined with MCA. Once again, the image is first split into low and high-frequency components, with rain fully contained in the high-frequency component. To do the separation into these components, possible rain pixels are detected by comparing the value of every pixel to the mean values of pixels in 7×7 patches such that it is the center of the patch and such that it is one of the corners of the patch. If the pixel is greater than all of these mean values, it is a potential rain pixel. The potential rain pixels are then replaced by the mean values of the patches that they are the center of. The image is then filtered using the guided image filter from [4]. This results in separating I_L that contains no rain and I_H that contains both rain and many of the image's fine details. Using the kSVD method on patches of images of real rain, the authors then learn a dictionary that can sparsely represent the rainy images. Then, all of the vectors in the dictionary are classified as belonging either to rain or non-rain components. First, the vectors with variance under a certain threshold are classified as non-rain vectors. Then, from the remaining vectors, those with average absolute horizontal gradient smaller than a second threshold are classified as belonging to non-rain vectors. And finally, for the remaining vectors, a histogram of oriented gradients is calculated. Using these histograms, we can calculate the dominant direction of the gradients in that vector. The assumption is that, since most of the remaining vectors are ones that describe rain, most of their dominant directions of the gradient will be similar. In order to filter out the remaining non-rain vectors, we can calculate the mean and variance of all directions. Those vectors with direction different

from the mean by significantly more than the variance are classified as non-rain vectors as well. The final image is then reconstructed using all the non-rain vectors from the learned dictionary.

2.2.4 Deep learning

Since the problem of single image rain removal is a more standard image denoising problem, most deep learning methods for rain removal are similar to other deep learning based denoising methods. They can be split into two main groups: supervised methods based on direct estimation of the clear image from the input rainy image, and autoencoder based methods. The concept behind the autoencoder based methods is to encode and then decode the input image and have the noise, in this case, rain, be eliminated in the compression during the process.

Progressive Image Deraining Network [32] is a convolutional neural network that progressively derains an image. The network works recursively by using its output as the input for the next step in rain removal. The network architecture contains five residual blocks made out of two convolutional layers and a ReLU non-linearity with skip connections around each of them. Before and after the residual blocks are additional convolutional layers and activations for input and output. The input for each step is the output of the previous step concatenated with the ground truth clean image. In the first step, the input is simply the rainy image. The loss function used is

$$L = -\sum_{t=1}^{T} \lambda_t \text{SSIM}(x^t, x^{gt}) \tag{2.43}$$

Here T denotes the total number of steps. x^t is the derained image after t steps, and λ_t is the weight assigned to the error at step t. In the final model the authors used $T = 6$ and $\lambda_1 = \lambda_2 = ... = \lambda_5 = 0.5$ and $\lambda_6 = 1.5$. The result is that the image is slightly more visually derained at every step, with the image being fully rain free at T steps. This kind of recursive approach is interesting as it allows the network to use fewer parameters, making the entire process less memory intensive.

Restormer [33] is a transformer for image denoising. The same architecture works well on multiple problems, image denoising, motion blur removal, out-of-focus deblurring, and rain removal. The architecture is similar to general vision transformers, with a few important differences. The biggest one is that the self-attention mechanism is pooled across channels rather than across spatial dimensions. This allows the transformer to have better local modeling power. Another important trick that is used is one change in training. The model is trained on larger batches of small image patches in the early training epochs. This allows the model to learn general local characteristics of the noise it is removing. During training, the size of the patches is progressively increased until the final epochs when the model is being trained on entire large images. This progressive increase in size allows the model to better learn to link the global context to the local features it learns in early epochs of training. The model is trained using

supervised learning and uses l_1 loss.

MCWNet [34] is a convolutional neural network for rain removal based on the U-net architecture. There are three downsamplings and three upsamplings. After each downsampling, as well as after the input and before downsampling, the network has two blocks consisting of a 3×3 convolutional layer and a non-linearity repeated three times. After each repetition, there is also a skip connection concatenating the initial input with the output of the previous layer. After these two convolutional blocks, there is a regional non-local block and then the downsampling or upsampling. The main addition to the standard U-net formulation is that the skip connections that are often present in U-nets do not only connect the same scale but are instead fully globally connected. This is accomplished by using additional fusing blocks. These blocks first perform global average pooling, then a fully connected layer, then a ReLU, then another fully connected and then a sigmoid. This output is elementwise multiplied with the inputs to every scale of the decoder. Formally this can be written down as

$$\begin{aligned} D^l_{concat} &= \left(\bigoplus_i H^i_l(E^i_{out})) \oplus H_{up}(D^{l+1}_{out}) \right) \\ D^l_{in} &= W_{1\times 1}\left(f_{fusion}(D^l_{concat}) \right) \end{aligned} \tag{2.44}$$

Here D^l_{in} is the input into the lth level of the decoder part of the U-net, E^i_{out} is the output of the ith level of the encoder, H_{up} is the upsampling operator, H^i_l is the downsampling operator from scale i to scale l, and $\oplus$ is the channel-wise concatenation operator. $W_{1\times 1}$ is a one by one convolution and f_{fusion} is the fusion block described above. The loss function used is

$$L = ||x^{gt} - f(x^{gt})||_1 + ||x^{gt} - f(x^{gt})||_2^2 \tag{2.45}$$

here, f denotes the entire neural network. The skip connections connecting across multiple scales enable this method to use features extracted on higher scales better on lower scales, resulting in better rain streak removal.

An example of an autoencoder for image deraining is **Variational image deraining** [35]. This method is based on a conditional variational autoencoder. The encoder network is used to estimate μ and σ vectors of the latent distribution, and the decoder network input is sampled from that distribution. In order to ensure that the estimated distribution is similar for a clear and rainy image pair, an additional subnetwork is used called a prior network. The prior network takes as input the clear version of the rainy image that is given to the encoder. This prior network is used as an additional regularizer of the conditional variational autoencoder. This is done by taking the KL divergence of the estimated distributions of the encoder and the prior network.

Chapter 3

Datasets

There are several types of datasets available that include weather degradation. The most commonly used ones are synthetic datasets constructed from clear images by artificially adding degradation, using models such as those described in equations (1.20) and (1.22). These datasets have the advantage that they also include a ground truth version of each image and so can be used for supervised learning and for full reference evaluation. However, they also have the disadvantage that they have synthetic rather than real images. There is also a number of datasets that have only hazy images, but evaluating on a large dataset without available ground truths is difficult, so they are mostly smaller datasets used for visual inspection. Finally, there are also task-driven datasets, which include additional information in the image, such as object detections, scene segmentation, or similar metadata.

Many of these datasets are constructed by choosing specific frames from long video recordings. Sometimes from dashboard cameras in cars, sometimes from surveillance cameras. This creates a problem for rain-degraded images, as single frames of these videos often do not contain rain streaks. This is because the length of the rain streak is proportional to the shutter speed of the camera, and image capture times are extremely low in modern video cameras. Because of this, there are very few datasets that include real rain streaks in them.

3.1 RESIDE

The RESIDE dataset was first introduced in [36]. It consists of several parts: synthetic images with ground truth available and real hazy images with detection annotations available.

The first two parts are the Indoor Training Set (ITS) and Outdoor Training Set (OTS). ITS consists of 110500 hazy images of indoor scenes. The haze is synthetically added using equation (1.20) to images with known depth maps. Ten images are generated from each ground truth image with varying values of β, resulting in different choices of t and A. The clear images come from the NYU2 dataset [37] and Middlebury stereo dataset [38]. Values of A are

Figure 3.1: Examples of clear and hazy images from the RESIDE dataset. The top row is an example from the indoor set, and the bottom row is an example from the outdoor set.

uniformly randomly distributed in the interval $[0.7, 1]$ with the assumption that airlight is fully uniform across the image and that it is gray. Values for β are uniformly sampled from the interval $[0.8, 1.6]$ again with the assumption that β does not vary across the image. The OTS consists of 313950 images generated in the same way, but based on outdoor images from datasets such as cityscapes [39]. The purpose of these two parts of the dataset is to train models in a supervised way since every image has a matching ground truth haze-free image. Figure 3.1 shows an example of images in the ITS and OTS.

The third part is called Synthetic Objective Testing Set (SOTS). It consists of 1000 synthetically generated hazy images. 500 of these are indoor images from the $NYU2$ dataset, and 500 are generated from cityscapes. Every clear image is only used to generate one hazy image, and the clear images used in this subset are not used in ITS and OTS. The purpose of this part of the dataset is to be able to test the accuracy of dehazing methods based on full reference metrics.

The final part of the dataset is the Real-world Test-driven Testing Set (RTTS). This part consists of 4322 real-world hazy images with object detection annotations. There is a total of 41203 annotated objects across the entire dataset divided into 6 classes: person, bike, car, bus, truck, and motorcycle. Most of the scenes in this dataset are scenes of traffic on roads or streets in cities. The dataset is designed as a testing set of dehazing methods on real hazy images by comparing the quality of detection on hazy and dehazed images. Figure 3.2 is an example of an

Figure 3.2: An example of an image from the RTTS section of RESIDE dataset. On the left is the image, in the middle is the image with ground truth bounding boxes shown for all objects, and right is the detections made in the hazy image.

Figure 3.3: An example of a problematic image from the RTTS dataset. The image itself is not very hazy, and multiple people are detected using an object detector that are not present in the annotations. The left is the original image, the middle is the image with ground truth bounding boxes, and the results of detections on the hazy image are on the right.

image from this part of the dataset. Note that only one of the cars is detected when the detector is applied directly to the hazy image. However, this dataset also has some problems. Some of the images in it are not very hazy, and some have poor annotations. The non-hazy images are actually interesting, as they can be used to test how much dehazing would degrade a scene with already good visibility. However, the non-annotated objects are a problem. Because of this, the dataset is not suitable for testing using mean average precision, as there will be correct detections marked as false positives. In order to avoid this problem, we use mean average recall, only measuring what percentage of objects were detected, rather than measuring false positives as well. Since dehazing should not increase false positives, this is a good measure of improvement in object detection. Figure 3.3 shows an example of a problematic image from the dataset. It is not particularly hazy, and the detector detected more people than are annotated in ground truth bounding boxes.

3.2 DAWN

Detection in Adverse Weather Nature (DAWN) is an adverse weather dataset introduced in [40]. The dataset consists of four types of weather degradation: fog, rain, snow, and sand. All the images in the dataset are real degraded scenes with detections annotated for the same classes as in RTTS. Since the rain images are captured using a high shutter speed camera, rain streaks are not visible in them, so we will only use the images of fog for our purposes. There is a total

Figure 3.4: Example of an image from the DAWN dataset. On the left is just the image. In the middle is the image with ground truth bounding boxes marked. On the right is the image with detections from a pre-trained detector. Note that the second car is not detected.

of 114 hazy scenes, mostly of roads and streets. Figure 3.4 is an example of an image from the DAWN dataset. The middle image is the ground truth bounding boxes, and the right is the detections using a pre-trained object detector. Note that in the detections, one of the cars is not detected.

3.3 Dense haze dataset

In [41, 42, 43], as a part of the CVPR dehazing challenge [44], authors created datasets of natural hazy images with approximate ground truth pairs. In order to do that, they used a fog-generating machine and took pictures with a static camera before and after generating fog. These datasets contain both indoor hazy images and outdoor images in parks and similar public spaces. Additionally, ColorCheckers are placed in every picture to make the estimation of dehazing quality even easier. The big problem with these datasets is that they are made using artificially generated fog, which does not scatter and attenuate the light in the same way natural fog does. While different wavelengths of light scatter differently in every type of haze, this is especially pronounced in this artificial fog. As a result, the approximation that t is constant across channels does not hold for images in this dataset, so they are not suitable for comparison of methods that make this approximation. It is important to note that the approximation of constant transmission in all channels is relatively good in images with natural haze, however.

In order to prove that these images do not conform to the model (1.20), we have tried to minimize the following expression

$$||I - tJ + (1-t)A||_2 \tag{3.1}$$

under the assumption that A is constant across the entire image. The result is a set of $3 * p$ equations with $p+3$ unknowns, where p is the number of pixels in an image, and the 3 comes from the number of channels. Note that here both I and J are known. The minimized expression gives t and A, which can be used to reconstruct a version of the dehazed image which contains

full detail, but has greatly reduced saturation in all pixels. This is because the effects on channels are averaged out by assuming that t is constant. An example of this can be seen in Figure 3.5.

Figure 3.5: Comparison of hazy image (left), clear image (middle), and image dehazed using the haze model and the known ground truth (right). The dehazed image has greatly reduced saturation, despite using the known ground truth. This indicates that the image does not satisfy the model very well.

3.4 FADE dataset

The FADE dataset, introduced in [45], is a dataset of a small number of real hazy images. The images have no ground truth, object detection annotations, or any such additional information. This dataset's primary purpose was to evaluate image dehazing using reference-free metrics, such as the one introduced in the same paper. Some examples of images from this dataset can be seen in Figure 3.6 shows examples of images from the FADE dataset.

Figure 3.6: Two examples of images from the FADE dataset.

Figure 3.7: Example of an image from the Rain100L dataset. The image on the right is generated from the image on the left.

3.5 Rain100L

Rain100L [46] is a dataset containing 300 image pairs, 200 for training, and 100 for testing purposes. Each image pair contains a clear image and a synthetically generated rainy image. The rain is generated using a slight modification of the model described in equation (1.22). The model used in this dataset is

$$I = J + LR \tag{3.2}$$

Here the added term L is a localization binary map with value 1 where rain streaks exist and 0 where they do not. Functionally, this is equivalent to the model used in equation (1.22), but with less constraints on R. In this variant of the dataset, all of the rain streaks have a similar direction of rainfall for a single image, making it more realistic than datasets which include many different directions in a single image. Examples of image pairs from the dataset can be seen in Figure 3.7. Due to the size, this dataset is usually only used for fine-tuning pre-trained models and testing trained models.

Figure 3.8: An example of an image from the Rain14k dataset. The image on the right is generated from the image on the left.

3.6 Rain14k

Rain14k [47] is a dataset that contains a total of 14000 rainy images generated from 1000 clear images. Each image is used to generate 14 images. The dataset is split into training and testing subsets such that no images from the testing subset are generated from the same image as those from the training subset. A downside of this dataset is that the rain generation is done using Photoshop, so the method for adding rain and the exact model of rain degradation used is not freely available. Examples of images from this dataset can be seen in Figure 3.8.

Figure 3.9: An example of an image from the real rain dataset. The image on the left is constructed from the video of the rain. The image on the right is a still frame from the video.

3.7 Real rain dataset

The realistic rain dataset [48] is a dataset that contains 300 videos of still scenes during rain. The authors proposed a way to estimate the background of the scene from these videos. This estimate of the rain-free scene is used as ground truth for single image rain removal on any still frame of the video. In theory, this could be used to generate many images, but in practice using more than one frame from each video for testing can skew the results. Examples of the still frames and derained versions of some images can be seen in 3.9.

Chapter 4

Single image dehazing using a smooth approximation of dark channel prior

4.1 Error analysis of single image dehazing

Since many methods for single image dehazing are based on first estimating transmission and airlight and then reconstructing the original image using those estimates, it is useful to analyze how errors in the estimates propagate. Throughout this section, ground truth transmission, airlight, and the haze-free image will be denoted by t, A, and J, while the estimates will be denoted by $\hat{t}$, $\hat{A}$ and $\hat{J}$. The estimate of reconstruction is calculated as

$$\hat{J} = \frac{I - \hat{A}}{\hat{t}} + \hat{A} \tag{4.1}$$

We can now use equation (1.20) to replace I and get the relation of J and $\hat{J}$.

$$\begin{aligned} \hat{J} &= \frac{I - \hat{A}}{\hat{t}} + \hat{A} \\ \hat{J} &= \frac{Jt + (1-t)A - \hat{A}}{\hat{t}} + \hat{A} \\ \hat{J} &= J\frac{t}{\hat{t}} + \frac{A - \hat{A}}{\hat{t}} - \left(\frac{t}{\hat{t}}A - \hat{A}\right) \end{aligned} \tag{4.2}$$

We can break down the analysis of this relationship based on the following assumptions.

•If we assume we know the exact value of airlight ($\hat{A} = A$)

•If we assume we know the exact value of transmission map ($\hat{t} = t$)

•If we assume that we have to estimate both of those values

If we assume that the exact value of airlight is known, the estimate becomes

$$\hat{J} = J\frac{t}{\hat{t}} + \left(1 - \frac{t}{\hat{t}}\right)A \tag{4.3}$$

Figure 4.1: An example of the effects of overestimating and underestimating the transmission map t. The top left is the original clear image with known depth for each pixel. The top right is a hazy image made using equation (1.20) and depth information of the image. The bottom left is the image reconstructed by underestimating t by 25% at every pixel, while the bottom right is made by overestimating by the same amount.

What we can note here is that the estimated reconstruction can be seen as a hazy version of the ground truth image, with the transmission map being the ratio $\frac{t}{\hat{t}}$. This is interesting, as it means that if $\frac{t}{\hat{t}} < 1$, the effect is simply not removing haze fully, however, if $\frac{t}{\hat{t}} > 1$, the effect can lead to oversaturation in those parts of the image. The simplest way to see this is by looking at it geometrically in the RGB space, where $t = 0$ means that the value of the pixel is fully defined by A, the value of $t = 1$ means that it is fully defined by the clear image, and values between 0 and 1 form a line between the two points. However, values larger than 1 push the point further along that line in *RGB* space, potentially moving it out of the unit cube, thus causing oversaturation. That leads to values greater than 1 or smaller than 0 in some channels, but since images are always clipped into the interval $[0,1]$, this will cause a loss of information. That potential loss of information leads us to conclude that it is better to overestimate $\hat{t}$ than it is to underestimate it. Figure 4.1 shows an example of the effects of overestimation and underestimation of transmission. The example is made using an image with a synthetically made hazy image, so the exact parameters t and A are known. Note how parts of the top right corner of the image made by overestimating are completely black.

Similarly, if we assume that the exact transmission is known, the equation becomes:

$$\hat{J} = J + \left(A - \hat{A}\right)\left(\frac{1}{t} - 1\right) \tag{4.4}$$

The error term here is amplified in the deeper parts of the image. Since the error is proportional to $\frac{1}{t}$, even a very small error in the estimate $\hat{A}$ can result in a significant error in $\hat{J}$. Because of this, we can conclude that it is reasonable to always use an approximation of t greater than some fixed $\varepsilon > 0$ to ensure that the error term coming from $\hat{A}$ does not explode in the deeper parts of the image. We can also note that the effect of error in $\hat{A}$ is symmetrical, so there is no need to bias it either way, unlike the effect of $\hat{t}$.

Finally, we can analyze the most realistic case, in which the exact t and A are both unknown. Since equation (4.3) indicated that the best way to model the error in t is multiplicative, we can write $\hat{t} = t(1 + e_t)$, and from equation (4.4) we know that error in A is modeled as additive we write $\hat{A} = A + e_A$. By putting these expressions into equation (4.2), we get the following expression

$$\begin{aligned} \hat{J} &= \frac{t}{t+e_t}J + \frac{-e_A}{t+e_t} - \left(\frac{t}{t+e_t}A - A - e_A\right) \\ &= J - \frac{e_t}{t+e_t}J - \frac{e_A}{t+e_t} - \left(\frac{-e_t}{t+e_t}A - e_A\right) \end{aligned} \tag{4.5}$$

If we now assume that our approximations $\hat{t}$ and $\hat{A}$ are reasonably good, in other words, that e_t and e_A are close to 0, we can analyze the error term

$$\frac{e_t}{t+e_t}J - \frac{e_A}{t+e_t} - \left(\frac{-e_t}{t+e_t}A - e_A\right) \tag{4.6}$$

In the parts of the image close to the camera, where $t >> e_t$, we can use the approximation $\frac{e_t}{t+e_t} \approx 0$. The expression for the total estimation error then becomes

$$\hat{J} - J \approx -\frac{e_A}{t+e_t} + e_A \tag{4.7}$$

So assuming that $t >> e_A$ as well, the entire error in $\hat{J}$ can be approximated to just the error in airlight estimation e_A.

To sum up, the conclusions we can take from the error analysis are:

•It is better to overestimate $\hat{t}$ than to underestimate it.

•Estimation $\hat{t}$ should be bounded away from 0.

•Error in $\hat{A}$ is best modeled as additive, while in $\hat{t}$ as multiplicative.

4.2 Variational formulation of dark channel prior for single image dehazing

The dark channel prior is one of the most successful priors for single image dehazing. One of its greatest problems, however, is the fact that it is stated as an explicit prior. To address this, we can first define the following function:

$$F(x) = \min_{c\in\{R,G,B\}} \min_{y\in\Omega_x} I_c(y) \tag{4.8}$$

Here the set Ω_x denotes the patch around x.

This function has value 0 if the point x satisfies the dark channel prior and a value greater than 0 if it does not. That allows us to discuss the extent to which a certain image satisfies the dark channel prior at any point. The natural extension is to define the total adherence of an image I to the prior as

$$E(I) = ||F||_2^2 \tag{4.9}$$

One big problem with this definition of the function F, and by extension E, is that it is not smooth. Both the localization and choosing the minimal value are nonsmooth functions and so cannot be used as a part of a loss function.

First, we will define $m(x)$ as the minimal channel of the image I

$$m(x) = \min_{c\in\{R,G,B\}} I_c x \tag{4.10}$$

Using this notation, the dark channel prior can be written as

$$\min_{y\in\Omega_x} m(y) \approx 0 \tag{4.11}$$

For a smooth approximation of finding a minimum, we will use the p norm. It is well known that

$$\lim_{p\to\infty} \sqrt[p]{\int f^p} = \lim_{p\to\infty} ||f||_p = \max f \tag{4.12}$$

Since this is a maximum, and we need a minimum, we will use the following identity

$$\min_x f(x) = 1 - \max_x (1 - f(x)) \tag{4.13}$$

This holds true for all functions with range bounded to the interval $[0,1]$. In order to find the minimum only in the neighborhood of x Ω_x, we will use convolutions. Specifically, by convolving the function m with a localizing function w. The most simple example of such a function is the characteristic function of set Ω. By calculating the convolution of p-th powers

of both w and m and then taking the p-th root of the result, we simulate a p-norm of only that specific part of the function. This, in combination with equations (4.12) and (4.13) gives us a smooth approximation of F from equation (4.8):

$$F_{smooth}(x) = 1 - \sqrt[p]{\int (1-m)^p * w^p} \tag{4.14}$$

Now we can write a smooth approximation of the dark channel prior of the entire image as

$$E_{DCP}(m) = \left|\left| \sqrt[p]{(1-m)^p * w^p} - 1 \right|\right|_2^2 \tag{4.15}$$

We can estimate the minimal channel of the dehazed image and the transmission map by minimizing the following functional:

$$L(m,t) = \lambda_1 ||I - mt - (1-t)A||_2^2 + \lambda_2 ||\nabla t||_1 + \lambda_3 ||t-1||_2^2 + \lambda_4 E_{DCP}(m) \tag{4.16}$$

The first term is the data term, which ensures that the dehazed image will still have the same features as the hazy image. The second term ensures the smoothness of the transmission map. It is reasonable to assume that the transmission map will not have texture in it, due to its exponential relation to the depth map. Here we use the 1-norm as it allows for less blurred edges. The third term biases the model to overestimate the transmission. This is done to ensure that oversaturation does not happen in reconstruction, as discussed in section 4.1. The fourth term is the smooth approximation of the dark channel prior, as discussed above.

One interesting theoretical advantage this formulation has over the standard DCP is that there is no assumption of constantness of t within a patch. This means that there is no need for postprocessing of the transmission, but it also allows for different sizes of the localization function w. Increasing the size of the support of w, and by extension the size of Ω_X, the underlying prior of DCP becomes more satisfied, however the term also becomes less informative.

The estimation of the transmission map is done by minimizing the functional L and taking

$$\hat{t} = \operatorname{argmin} L(m,t) \tag{4.17}$$

The airlight $\hat{A}$ is estimated using the same method as described in the original Dark Channel Prior paper [3]. In order to find the minimum, we need the gradient of each term, including E_{DCP}. This gradient is given by the expression:

$$\delta E_{DCP}(m) = -2\left(\sqrt[p]{(1-m)^p * w^p} - 1\right) \frac{(1-m)^{p-1} * w^p}{\sqrt[p]{((1-m)^p * w^p)^{p-1}}} \tag{4.18}$$

For large values of parameter p the p norms converge to ∞ norm. Because of this, we have the

approximation $||\cdot||_p \approx ||\cdot||_{p-1}$. This can be used in equation 4.18 by viewing the fraction as a ratio of p norm and $p-1$ norm of the localized version of the function $1-m$, taken to the power $p-1$. Since the p norms are monotonous in p, the fraction is a number close to 1 but greater than 1. Taking a p-th root of this number for a large p allows us to make the approximation

$$\frac{(1-m)^{p-1} * w^p}{\sqrt[p]{((1-m)^p * w^p)^{p-1}}} \approx 1 \tag{4.19}$$

The gradient can then be approximated as

$$\delta E_{DCP}(m) \approx 2\left(\sqrt[p]{(1-m)^p * w^p} - 1\right) \tag{4.20}$$

This approximation can be useful for speeding up gradient descent.

The optimization of the functional is done using coordinate descent, alternating steps in the direction of the gradient with respect to m and with respect to t. The expression for gradient of E_{DCP} is written down in equation (4.18), but it is still necessary to find the gradient of the other terms in the functional. The gradients of $||I-mt-(1-t)A||_2^2$ and $||t-1||_2^2$ are simple, as these functions are smooth with respect to m and t. The gradient of the term $||\nabla t||_1$ is more complicated, as the 1-norm is not smooth. We can approximate the absolute value with the function

$$f(x) := \sqrt{x^2+\varepsilon^2} \approx |x| \tag{4.21}$$

Since the argument of the square root is always greater than 0 for $\varepsilon > 0$, the function f is smooth. From the Euler-Lagrange equations 1.7 we know how to calculate the gradient with respect to partial derivatives t_x and t_y. So the approximate gradient of the term $||\nabla t||_1$ is the function s defined as:

$$s(t) = \frac{\varepsilon^2}{\sqrt{\varepsilon^2+t_x^2}} t_{xx} + \frac{\varepsilon^2}{\sqrt{\varepsilon^2+t_y^2}} t_{yy} \tag{4.22}$$

The full description of the algorithm is given in Algorithm 1. Both m and t are kept within the constraint of $0 \leq m,t \leq 1$ using a projectional method in each iteration.

When defining iterative algorithms, it is always important to discuss the question of convergence. Let (m_0,t_0) be the starting point of the algorithm. The sequence (m_n,t_n) is defined recursively as

$$\begin{aligned} m_{n+1} &= m_n - dt \cdot \frac{\partial L(m_n,t_n)}{\partial m} \\ t_{n+1} &= t_n - dt \cdot \frac{\partial L(m_{n+1},t_n)}{\partial t} \end{aligned} \tag{4.23}$$

Where L is the functional defined in equation (4.16). Since each step is a gradient descent step for one of the parameters, we know that the value of the functional is decreasing. If we define

Algorithm 1 Variational Dark Channel Prior algorithm

$\hat{t} = 0$
$\hat{m} = m$
while $\left(d_m^2 + d_t^2 > tol\right)$ **do**
$$\hat{m}_{new} = \hat{m} - dt\Big[\lambda_1\left(\hat{m}\hat{t} + (1-\hat{t})\hat{A} - m\right)\hat{t} - \lambda_4 \delta E_{DCP}(\hat{m})\Big]$$
$$\hat{t}_{new} = \hat{t} - dt\Big[\lambda_1\left(\hat{m_{new}}\hat{t} + (1-\hat{t})\hat{A} - m\right)(\hat{m}_{new} - A) + \lambda_2 s(\hat{t}) - \lambda_3(t-1)\Big]$$
$d_m = ||\hat{m} - \hat{m}_{new}||_2$
$d_t = ||\hat{t} - \hat{t}_{new}||_2$
$\hat{m} = \hat{m}_{new}$
$\hat{t} = \hat{t}_{new}$
end while

Symbol	Meaning
$\hat{t}$	Estimated transmission map
$\hat{m}$	Estimated minimal channel of dehazed image
m	Minimal channel of the input image
d_m	Change in $\hat{m}$ in previous step
d_t	Change in $\hat{t}$ in previous step
tol	threshold of change for stopping the algorithm
λ_1, λ_2, λ_3	model hyperparameters

Table 4.1: Explanations of the meaning of all variables in algorithm 1.

$L_n := L(m_n, t_n)$ we can see that

$$L_n = L(m_n, t_n) > L(m_{n+1}, t_n) > L(m_{n+1}, t_{n+1}) = L_{n+1} \tag{4.24}$$

Since $L \geq 0$, the sequence L_n is a decreasing sequence bounded from below, so it is convergent. Due to the definition of m_n and t_n, we can see that the limit set of these sequences is a set of local minimums of L. This can be a single point if L has an isolated minimum, or it can be a manifold of connected local minimums. We also know that $0 \leq m_n, t_n \leq 1$ for every component of m_n and t_n, so both of these sequences are defined on a compact set. Then the sequence $(m_n, t_n)_n$ is also a sequence defined on a compact set. Because of that, this sequence must have a convergent subsequence, and that subsequence converges to a local minimum of L. As the subsequence

converges, the difference between consecutive elements is bounded by

$$\begin{aligned} d_m &\leq \left|\left| dt \frac{\partial L(m_n, t_n)}{\partial m} \right|\right|_2 \\ d_t &\leq \left|\left| dt \frac{\partial L(m_{n+1}, t_n)}{\partial t} \right|\right|_2 \end{aligned} \tag{4.25}$$

Since the partial derivative converges to 0 as we approach a local minimum, so do d_m and d_t. Because of this, for a small enough value of dt, the algorithm will always terminate near a local minimum of the functional L.

4.3 Unsupervised image dehazing using a smooth approximation of dark channel prior

Using variational methods can be slow, as they need to minimize a complicated functional for every image they are used on. However, variational methods can, in general, be easily adapted into unsupervised methods. The simplest way to do this is to train a neural network using the functional defined in the variational method as a loss function. This can have both advantages and disadvantages. The main advantages are in the speed of inference once the network is trained, as even large networks usually have lower inference time than fully minimizing a variational method. Another potential advantage is the freedom to estimate more parameters. In the case of dehazing, our variational method is incapable of estimating A simultaneously with other parameters, as that would cause too many local minimums to appear and could cause the method to converge to a bad reconstruction. However, a neural network is trained on a large number of images, so they can learn to estimate a minimum even with this increased number of parameters. A disadvantage is that there is no guarantee of the network generalizing, and we do not have a mathematical guarantee that its output is a minimum of the functional.

We propose an unsupervised single image dehazing method based on the following loss:

$$\begin{aligned} L(m,t,A) =& ||I - mt - (1-t)A||_2^2 + \lambda_1 ||\nabla t||_1 + \lambda_2 ||t-1||_2^2 \\ &+ \lambda_3 \text{Var}(A) + \lambda_4 ||\text{blur}(A) - A||_2^2 + \lambda_5 E_{DCP}(m) \end{aligned} \tag{4.26}$$

The first four terms are the same as in the loss function in equation 4.16. The additional terms that contain A are important since this method jointly estimates A along with t and m. The term $\text{blur}(A)$ is defined as a convolution of A with a 5×5 kernel of constant values $\frac{1}{25}$. This term serves to ensure that the estimated A is smooth. The term $\text{Var}(A)$ is the variance of A defined as

$$||A - \text{mean}(A)||_2^2 \tag{4.27}$$

The purpose of this term is to ensure that the estimated value of A is close to a constant across the entire image.

The fact that airlight A is allowed to be non-constant is a major advantage over many other methods. While A being constant is a common assumption, it is not always correct in real images. This can cause problems, especially in parts of the image that are further from the camera, as discussed in the error analysis section.

The architecture of the network used for this approach is a U-net. It consists of 5 down-sampling blocks and 5 upsampling blocks with skip connections in the lowest two blocks. Each block consists of a 3×3 convolution, batch normalization, and ReLU non-linearity followed by a 1×1 convolution, batch normalization, and ReLU non-linearity. An image sketching the architecture is shown in Figure 4.2.

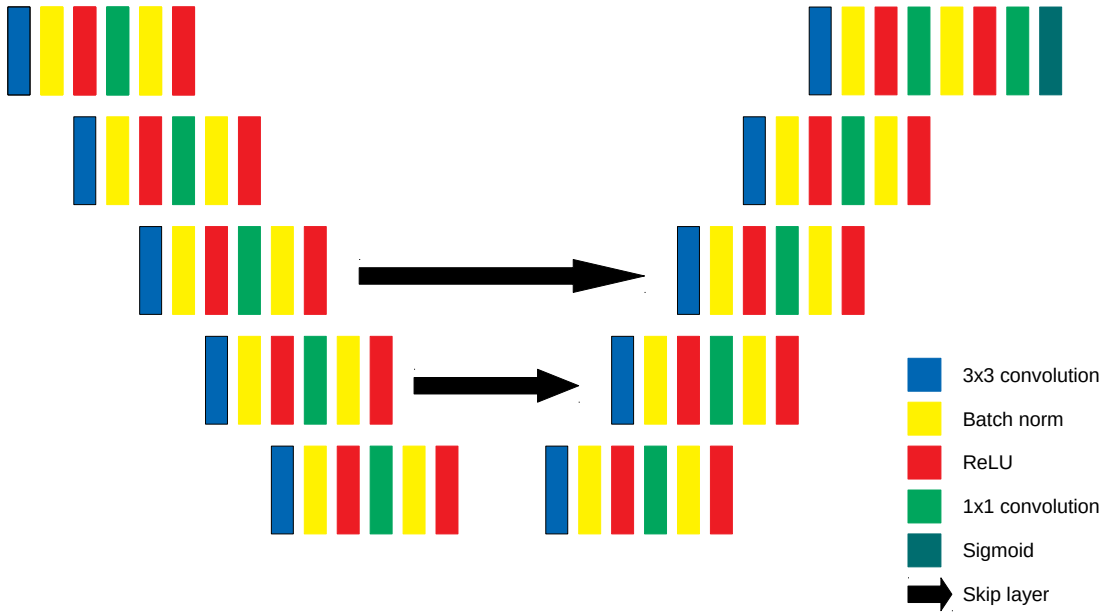

Figure 4.2: A sketch of the architecture used in the unsupervised learning method.

The training is done by simultaneously training three copies of this network, one that estimates $\hat{m}$, one that estimates $\hat{t}$, and one that estimates $\hat{A}$. All three networks are updated simultaneously. For the purposes of the inference, we only use the t and A networks and then estimate $\hat{J}$ using the outputs of these two networks. The training results are better when using the approximation defined in equation (4.20) for the gradient of E_{DCP} than when using the full gradient. This is because the approximation is more numerically stable and prevents issues with the gradient.

The setup of this unsupervised method is very similar to that of the double DIP [14] described in the prior work section 2.1.2. In both cases, there are three networks generating the clear image, the transmission map, and the airlight. The main differences are that we use more

regularizers to guide the optimization to converge to the minimum we want and the fact that our method is trained rather than optimized for every single image. This training reduces the time needed for inference by several orders of magnitude.

Chapter 5

Rain removal using weighted directional total variation

Throughout this chapter we will denote the input image with S rather than I as in other chapters. This is done because I can also mean the identity matrix, which is used in many equations and could cause confusion.

5.1 Automatic detection of the angle of rainfall

Detecting the angle at which the rain is falling is an important step in our method of removing rain streaks. The angle determines in which direction we demand the image to be smooth, meaning that an incorrect estimation of this parameter can lead to not removing the rain properly, as well as removing other edges from the image. The method for angle detection is based on the deraining method described in [23]. A major problem with that method is that it often does not find all of the rain streaks in the image. However, for purposes of estimating the angle, it is enough to find a large number of rain streaks, and then we can estimate that most of the rain is falling in a similar direction. The main assumption that this method makes is that rain streaks are generally brighter than the scene and that they are longer than they are wide.

The first step is to detect which pixels are brighter than their environment. This is done by comparing the brightness of the pixel to the average brightness of five patches, one centered around it, and the four where the target pixel is one of the corners. If the brightness of the pixel is larger than the average of all four of these patches, the pixel is considered to potentially belong to a rain streak.

After every pixel has been classified as either potentially rain streak or not rain streak, we perform the morphological operation closing and then find connected components of the resulting image. Each connected component can be seen as a set of 2D vectors, the coordinates of the pixels in those components. For each component we perform PCA, resulting in two principal

components, p_1 and p_2 and their respective values v_1 and v_2. The assumption we make is that many of the connected components we found this way are individual rain streaks, so we will use assumptions about rain streaks to filter out the connected components that do not represent rain, and then we will use the components to determine the angle.

The first assumption we will use is that rain streaks are much longer than they are wide. In other words that they have an extreme aspect ratio. The aspect ratio can be calculated using the ratio of principal values v_1 and v_2

$$r = \frac{v_2}{v_1} \tag{5.1}$$

A small r indicates a more extreme aspect ratio. For our purposes, we simply eliminate all connected components with an aspect ratio larger than some threshold t_r. Then, for the remaining connected components, we calculate their orientation φ as

$$\varphi = \arctan\left(\frac{p_1[2]}{p_1[1]}\right) \tag{5.2}$$

We then discretize the φ values into 30 bins in the interval $[0, 2\pi]$. The mode of this discretized set is the estimated angle of rainfall.

5.2 Single image rain removal using weighted directed total variation

The main issue variational methods for single image rain removal suffer from is a lack of localization. The effects of rain on an image are local, contained only in parts degraded by rain streaks, so smoothing every part of the image will almost certainly smooth out details that we would like to keep. In order to avoid this, we introduce a new factor in our functional, the weight map W. This map encodes the estimated likelihood that a pixel is part of a rain streak, the higher the likelihood, the more smoothing is applied to that pixel. The main advantage that W brings is the ability to differentiate between vertical edges in an image and rain streaks that we want to filter out. The new functional looks as follows:

$$\min_{R \leq S} \lambda_1 ||W\nabla_\Phi(S-R)||_1 + \lambda_2 ||R||_1 + ||\nabla_{\Phi^\perp} R||_1 \tag{5.3}$$

The first term in the model penalizes gradient in the direction perpendicular to the movement of rain streaks, resulting in overall smoothing of the image in that direction in the minimizer. The second term ensures the locality of rain by promoting the sparsity of the solution. The third term similarly penalizes the removal of edges in the direction of rain streaks, since those edges are not created by the effects of rain. The meanings of all variables can be found in table

5.1. The weight matrix W can either be taken as a square diagonal matrix, if we assume that the image is a vector, or as a matrix of the same shape as the image, in which case the multiplication is done as a Hadamard product. The 1-norm is used in all three terms, as it promotes sparsity. Sparsity in gradient, also called total variation regularization, translates to smoothness with well-defined edges in the image. The directional gradient ∇_Φ is defined as:

$$\nabla_\Phi = \cos(\Phi) \cdot \nabla_x + \sin(\Phi) \cdot \nabla_y \tag{5.4}$$

Symbol	Meaning
S	Input image
R	Estimated rain layer
W	Weight matrix of rain detection
Φ	Angle of rainfall
λ_1	Weight of the first term
λ_2	Weight of the second term

Table 5.1: Names and meanings of all variables used in functional (5.3)

The proposed model is an improved version of the UGSM model proposed in [26]. One change is simply a technical detail. In the UGSM paper, the authors handle directional gradient by rotating the image. However, image rotation is inconsistently defined across different libraries and can cause problems near the edges of the image, where padding is done to keep the square shape of the rotated image. A bigger, and more important, change related to the angle of the rain is that in our proposed method, the estimation of the angle of rainfall is automatic, rather than a separate input. This makes the proposed method more usable in realistic scenarios. The biggest and most important difference is the addition of the weight matrix W. This allows for better localization of rain streak removal, preventing oversmoothing in parts of the image that are not affected by rain, and improving it in those that are affected by it.

Since the functional defined in equation (5.3) uses the nonsmooth 1-norm in all of its terms, it is difficult to find its minimum. In order to do it, we will transform the unconstrained optimization into constrained optimization as follows:

$$\begin{aligned} &\min_{u,v,w} \lambda_1 ||u||_1 + \lambda_2 ||v||_1 + ||w||_1 \\ &\textbf{such that } u = W\nabla_\Phi(S-R) \quad v = R \quad w = \nabla_{\Phi^\perp} R \quad R \leq S \end{aligned} \tag{5.5}$$

This expression can now be minimized using the augmented Lagrangian and the alternating directions method of multipliers as described in section 1.1.1. The augmented Lagrangian is

given in the form

$$\begin{aligned} L(R,u,v,w,p_1,p_2,p_3) = \lambda_1||u||_1 + \langle p_1, W\nabla_\Phi(S-R)-u\rangle + \frac{\beta_1}{2}||W\nabla_\Phi(S-R)-u||_2^2 \\ \lambda_2||v||_1 + \langle p_2, R-v\rangle + \frac{\beta_2}{2}||R-v||_2^2 + \\ ||w||_1 + \langle p_3, \nabla_{\Phi^\perp}R-w\rangle + \frac{\beta_3}{2}||\nabla_{\Phi^\perp}R-w||_2^2 \end{aligned} \tag{5.6}$$

In order to minimize this expression, we will cyclically minimize the expression for each of the variables while keeping the other variables constant. Due to convexity, this will converge to the minimum in the limit. The reason for solving for each variable independently instead of simultaneously solving for all of them is that each subproblem is much simpler to solve for. Every subproblem is either solved using the proximal operator for the 1-norm or is smooth. The smooth subproblems can be solved by setting the gradient of the system to 0 and solving the resulting system of linear equations.

The subproblem for u can be written as

$$\begin{aligned} &\underset{u}{\operatorname{argmin}}\, \lambda_1||u||_1 + \langle p_1, W\nabla_\Phi(S-R)-u\rangle + \frac{\beta_1}{2}||W\nabla_\Phi(J-R)-u||_2^2 \\ =&\underset{u}{\operatorname{argmin}}\, \lambda_1||u||_1 + \langle p_1, W\nabla_\Phi(S-R)-u\rangle + \frac{\beta_1}{2}||W\nabla_\Phi(J-R)-u||_2^2 + ||p_1||_2^2 \\ =&\underset{u}{\operatorname{argmin}}\, \lambda_1||u||_1 + \frac{\beta_1}{2}\left|\left|W\nabla_\Phi(S-R)-u+\frac{p_1}{\beta_1}\right|\right|_2^2 \end{aligned} \tag{5.7}$$

This minimization is of the form

$$\underset{x}{\operatorname{argmin}}\, \lambda||x||_1 + ||x-y||_2^2 \tag{5.8}$$

So it can be solved using the proximal operator of the l_1 norm:

$$\operatorname{shrink}(x,\lambda) = \operatorname{sgn}(x)\max(|x|-\lambda, 0) \tag{5.9}$$

This means u can is updated as

$$u \leftarrow \operatorname{sgn}\left(W\nabla_\Phi(S-R)+\frac{p_1}{\beta_1}\right)\max\left(W\nabla_\Phi(S-R)+\frac{p_1}{\beta_1}-\frac{\lambda_1}{\beta_1}, 0\right) \tag{5.10}$$

every iteration.

The updates for v and w can be similarly derived to be

$$v \leftarrow \operatorname{sgn}\left(R+\frac{p_2}{\beta_2}\right)\max\left(R+\frac{p_2}{\beta_2}-\frac{\lambda_2}{\beta_2}, 0\right) \tag{5.11}$$

for v and

$$w \leftarrow \operatorname{sgn}\left(\nabla_{\Phi^\perp} R + \frac{p_3}{\beta_3}\right) \max\left(\nabla_{\Phi^\perp} R + \frac{p_3}{\beta_3} - \frac{1}{\beta_3}, 0\right) \tag{5.12}$$

for w.

The R subproblem is smooth, since it only contains 2-norms. Since it is smooth, we can take the gradient with respect to R and set it equal to 0. This gives us a system of linear equations

$$\begin{aligned}\left(\beta_1 \nabla_\Phi^T W^2 \nabla_\Phi + \beta_2 I + \beta_3 \nabla_{\Phi^\perp}^T \nabla_{\Phi^\perp}\right) R = &\left(\nabla_\Phi^T W \left(\beta_1 W \nabla_\Phi S - \beta_1 u + p_1\right)\right) \\ &+ \beta_2 v - p_2 + \nabla_{\Phi^\perp}^T \left(\beta_3 w - p_3\right)\end{aligned} \tag{5.13}$$

Solving this system of equations gives us the updated value for R. The system is of dimension $n \times 2$ where n is the number of pixels in the image, and finding the inverse of a matrix has the complexity $O(n^3)$. For most reasonably sized images, this complexity is too large, so directly solving the system is not feasible. If we assume W is constant, the left-hand side can be seen as a convolutional operator, and the system can then be solved using the Fast Fourier Transform. This is how it is done in UGSM. However, when using a non-constant W, the left-hand side is not a convolution, so FFT cannot be used for a fast solution to the system. However, we can use the sparsity of all matrices in the expression

$$\beta_1 \nabla_\Phi^T W^2 \nabla_\Phi + \beta_2 I + \beta_3 \nabla_{\Phi^\perp}^T \nabla_{\Phi^\perp} \tag{5.14}$$

Both I and W are diagonal, and discrete gradient matrices ∇ only have several diagonals different than 0. This means that the action of the left-hand side on any vector can be calculated efficiently, in complexity $O(n)$, instead of the standard $O(n^2)$. That makes this system well suited for iterative solvers such as the Generalised Minimal Residual Method (GMRES). GMRES is an iterative method, with each update having the complexity of $O(n)$, and it converges in relatively few steps for most sparse problems. In our case, around 30 iterations is enough for convergence. This way, GMRES gives an approximate solution to the linear system (5.13) in $O(n)$.

Finally, updates for multipliers p_1, p_2 and p_3 are done as

$$\begin{aligned} p_1 &\leftarrow p_1 + \beta_1 \left(W \nabla_\Phi (S - R) - u\right) \\ p_2 &\leftarrow p_2 + \beta_2 \left(R - v\right) \\ p_3 &\leftarrow p_3 + \beta_3 \left(\nabla_{\Phi^\perp} R - w\right) \end{aligned} \tag{5.15}$$

The last question is one of choosing a suitable matrix W. We will discuss specific methods of constructing it in Section 6.2.1, while here we will discuss the desired properties of the matrix. First is the question of detecting too much or too little. If we simply set $W \equiv 1$, we get a method essentially the same as UGSM, meaning we get the same problems with oversmoothing edges

in the same direction as rain. This tells us that we should aim to have at least some areas with value W lower than 1. If we let $W \equiv 0$, the obvious minimum of the functional is not changing the input image at all. The first term is 0 no matter what we do, so only the second and third terms matter. The third term is 0 if R is constant in the direction of $\Phi^{\perp}$. The second term is the smallest if $R = 0$. Due to the discretized nature of images, this, in practice, means the minimum will be 0 everywhere. This means that the optimal choice of W is partially problem-dependent. If smoothing out edges that should exist in a rain-free image is a bigger problem than leaving some rain streaks in it, then W should have smaller values in general and should only have values close to 1 for pixels we are very sure are parts of a rain streak. If the existence of rain streaks is more problematic, then W should have more values close to 1.

The full description of the proposed method for single image rain removal using weighted directed total variation can be seen in algorithm 2.

Algorithm 2 Single image rain removal using weighted directed total variation

Take input image S
Calculate W
Calculate Φ as discussed in section 5.1
Set $d_R = 1$ and $\varepsilon = 10^{-3}$
Set $n_{iter} = 0$ and $n_{max} = 600$
while $d_R > \varepsilon$ and $n_{iter} < n_{max}$ **do**
  Update u as in described equation (5.10).
  Update v as in described equation (5.11).
  Update w as in described equation (5.12).
  $R_{old} = R$
  Update R using GMRES as described in equation (5.13).
  $d_R = \frac{||R_{old} - R||_2}{||R_{old}||_2}$
  Update p_1, p_2 and p_3 as described in equations (5.15).
  $n_{iter} = n_{iter} + 1$
end while
Return derained image $S - R$

Chapter 6

Experiments

In general, measuring the success of image restoration can be done in four ways. The first way is using full-reference metrics. The most used examples of these metrics are peak signal-to-noise ratio (PSNR) and structural similarity index (SSIM). These metrics require the ground truth to be known, which is why they are called full-reference, and measure the quality of the reconstruction using it. PSNR is defined as

$$PSNR(x,y) = 10\log_{10}\left(\frac{M^2}{\frac{1}{c\cdot n}\sum_i (x(i)-y(i))^2}\right) \tag{6.1}$$

The variables x and y are the two images being compared. Here M denotes the maximal possible value of the image. It is most commonly 1 if the images are normalized to range $[0,1]$, or 255 if images are in UINT8 format. The sum in the denominator goes over all the pixels and across all channels, c represents the number of channels, and n is the number of pixels. In this metric, higher numbers mean better results, and the values the metric takes are unbounded. The PSNR of two fully identical images is impossible to calculate, as the mean square error in the denominator would be 0, and thus result in an error in the division.

SSIM is defined as

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{6.2}$$

Again x and y are patches of the two images being compared, μ_x and μ_y denote means of the two signals, σ_x, and σ_y denote standard deviations of x and y and σ_{xy} is their covariance. The constants c_1 and c_2 are used to ensure numerical stability when the denominator has small values. Most commonly used values of these constants is $c_1 = 10^{-4}$ and $c_2 = 9 \cdot 10^{-4}$. The SSIM metric takes values between -1 and 1, with 1 being attained only by identical signals and -1 by signals with exactly opposite values at each point. In order to get the total structural similarity between the two images, we take the average of SSIM scores for all fixed-size sliding

windows across the image. These are most commonly 8×8 square sliding windows or 11×11 weighted Gaussian sliding windows.

Since they require the ground truth, these metrics are primarily used on synthetic datasets, where the degraded images are constructed from known ground truths.

The second way is using no-reference metrics. These metrics do not require the ground truth to be known and can be used in more cases. They are usually defined in a problem-specific way, as they only measure the presence or absence of a specific type of degradation rather than fully measuring the reconstruction quality. Because of this, these metrics make poor loss functions for training but potentially good evaluation functions.

The third way is task-driven comparison. This test how well image reconstruction models can be used as preprocessing for other computer vision tasks. The most commonly used tasks for these tests are image segmentation and object detection. It requires a dataset with annotations and so is not entirely no-reference. The tests can be done both with synthetic images and natural images, but natural images give a better demonstration of how well the methods work in the real world.

The fourth way is by simple visual inspection. While it is the most subjective way of evaluating methods, it can be helpful to see in which cases particular methods seem to work well and in which cases they seem to have problems.

6.1 Dehazing

In order to properly test how well our methods work, we need to compare them to state-of-the-art methods for the same task. In the case of single image dehazing, we will be comparing with Dehazeformer [16], a transformer trained for image dehazing, FFANet [17], an attention-based convolutional neural network, and MSCNN [49] a multi-scale convolutional neural network. These methods are considered state-of-the-art for single image dehazing.

6.1.1 Hyperparameter choices

The choice of hyperparameters can have a lot of influence on the results of an experiment. In both our proposed variational and self-supervised method we tune the λ_i parameters to ensure that each part of the loss function is of the same order of magnitude and does not dominate the entire loss. Table 6.1 shows the chosen values for parameters of Variational DCP method in all performed experiments. The chosen localization function, in our case, is the characteristic function of a square around the origin with the side of size r. The full algorithm is described in Algorithm 1.

Parameter	Value
λ_1	1
λ_2	1
λ_3	0.05
λ_4	0.1
dt	10^{-2}
tol	0.1
r	20

Table 6.1: Values of parameters used in the testing of Variational DCP method. λ are weights in the functional, dt is the step size of gradient descent, tol is the step size condition for terminating optimization and r is the size of the localization function w. The algorithm is described in Algorithm 1

.

The unsupervised DCP method was trained on the hazy images from RESIDE OTS. It was trained for a total of 2 epochs due to the size of the dataset and using the learning rate of 10^{-3} with ADAM optimizer. The values of all hyperparameters used are shown in Table 6.2. The network itself has an increasing number of channels after each downsampling block and decreasing after upsampling. The outputs of the downsampling blocks have $[8, 16, 32, 64, 128]$ channels, while the upsampling blocks have $[64, 32, 16, 8, 3]$ channels in output.

Parameter	Value
λ_1	$5 \cdot 10^{-3}$
λ_2	10^{-3}
λ_3	1
λ_4	1
λ_5	10^{-2}
epochs	2
lr	10^{-3}

Table 6.2: Hyperparameters used in training the unsupervised DCP method. Parameter lr denotes the learning rate of the ADAM optimizer that was used.

For comparison with our image dehazing models, we used state-of-the-art models from the literature. These are FFANet, Dehazeformer, AOD-Net, and MSCNN. The tests were all done on *NVIDIA RTX 3090* except for the variational DCP model, which was implemented on CPU. A comparison of memory efficiency and speed can be seen in table 6.3. The variational method has the lowest memory footprint, needing only to store the image and a small number of extra

parameters, but is also orders of magnitude slower than the neural network-based models. This might be partially explained by the different hardware used, but a large part is also due to the fact that the method has to perform a lot of costly optimization for each evaluation. Of the other methods, the proposed unsupervised method is the smallest and fastest while either outperforming or performing on par with the supervised methods.

Method	# of parameters	inference time
Variational DCP	0	196s
Unsupervised DCP	0.79M	50ms
Dehazeformer	2.51M	231ms
FFANet	4.45M	443ms

Table 6.3: Comparison of the number of parameters and inference speed of proposed methods and some state-of-the-art methods.

6.1.2 Full reference

For full-reference comparison, we used RESIDE ITS and OTS datasets. The test parts of these two datasets together are sometimes called the Synthetic Objective Testing Set (SOTS). SOTS is composed of synthetic hazy images with known ground truths. The PSNR and SSIM results are shown in table 6.4. The baseline in the table refers to performing no dehazing and is used to ensure that all of the methods actually improve the images rather than degrade them. Note that the state-of-the-art models significantly outperform our proposed models. This is reasonable since these models are all trained in a supervised manner on synthetic images, just like the ones used in these tests. For comparison, we also added the results of the dark channel prior method to show that, although our methods are based on the same underlying prior, they do outperform the DCP method. Examples of ground truth, hazy and dehazed images from these datasets can be seen in Figures 6.1 and 6.2. The poorer performance, especially on ITS, can be partially explained by the fact that the dark channel prior is not well satisfied by indoor images. The images that contain large featureless white patches, such as walls, in particular, do not conform to the prior at all and so often cause DCP-based methods to underperform. This is seen well in the discoloration of walls in Figure 6.1 in images dehazed using both the unsupervised and the variational methods.

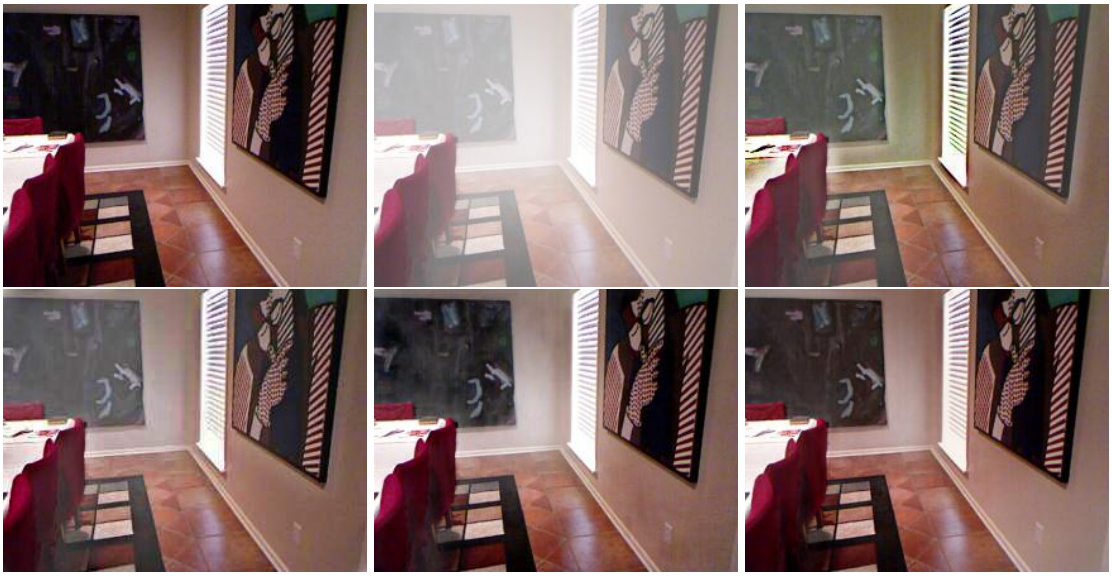

Figure 6.1: Example of an image from the indoor portion of the testing set. In the top row are the ground truth on the left, then the generated hazy image, and then the image dehazed using the Unsupervised DCP method. In the bottom row are images dehazed using the Variational DCP method on the left, FFANet in the middle, and Dehazeformer on the right. Here, the supervised methods heavily outperform the proposed methods.

Figure 6.2: Example of an image from the outdoor portion of the testing set. In the top row are the ground truth on the left, then the generated hazy image, and then the image dehazed using the Unsupervised DCP method. In the bottom row are images dehazed using the Variational DCP method on the left, FFANet in the middle, and Dehazeformer on the right. Here, the supervised methods heavily outperform the proposed methods.

Model	PSNR ITS	SSIM ITS	PSNR OTS	SSIM OTS
Baseline	11.98	0.7052	15.65	0.8289
Variational DCP	16.81	0.8056	21.89	0.9189
Unsupervised DCP	16.28	0.7540	19.47	0.8855
Dehazeformer	**40.05**	**0.996**	**34.95**	**0.9840**
FFANet	35.77	0.9846	33.57	0.9804
MSCNN	17.57	0.8102	23.16	0.9103
AODNet	19.06	0.8504	24.14	0.9200

Table 6.4: Table showing SSIM and PSNR of state-of-the-art models compared to our proposed models, Higher numbers indicate better restoration. Proposed models are written in bold letters. Baseline refers to performing no dehazing at all.

6.1.3 No-reference

The most widely used no-reference metric for dehazing is the FADE metric introduced in [45]. In order to calculate it, we first subdivide the image into patches and calculate 12 natural scene statistics for each patch. The list of these statistics and their descriptions can be seen in Table 6.5. The abbreviation MSCN refers to Mean Subtracted Contrast Normalized coefficients. This

ID	Description	Formula
f_1	Variance of MSCN coefficients	(6.3),(6.4)
f_2, f_3	The variance of vertical product of MSCN coefficients (positive and negative mode)	(6.3), (6.5)
f_4	Sharpness	(6.6)
f_5	Coefficient of variance of sharpness	(6.6)
f_6, f_7, f_8	Contrast energy in the patch (grayscale, yellow-blue, red-green)	(6.7)
f_9	Image entropy	(6.9)
f_{10}	Dark channel prior adherence	(4.8)
f_{11}	Saturation	(6.10)
f_{12}	Colorfulness	(6.11)

Table 6.5: List and description of the natural image statistics used in the calculation of the FADE metric.

is a natural image statistic calculated at each pixel using the formula

$$\begin{aligned} I_{MSCN}(i,j) &= \frac{I_{gray}(i,j) - \mu(i,j)}{\sigma(i,j)+1} \\ \mu(i,j) &= \sum_{k=-K}^{K}\sum_{l=-L}^{L} w_{k,l} I_{gray}(i+k, j+l) \\ \sigma(i,j) &= \sqrt{\sum_{k=-K}^{K}\sum_{l=-L}^{L} w_{k,l}\left(I_{gray}(i+k, j+l) - \mu(i,j)\right)^2} \end{aligned} \tag{6.3}$$

The $w_{k,l}$ denotes a 2D circularly symmetric Gaussian distribution used as a weighting function sampled out to 3 standard deviations and rescaled to unit volume. The calculation of f_1 inside a patch P then becomes the standard calculation of the variance of a vector:

$$f_1 := \frac{1}{n-1}\sum_{(i,j)\in P}(I_{MSCN}(i,j) - \mu_{MSCN})^2 \tag{6.4}$$

The vertical product of the MSCN coefficients refers to

$$I_{VProd}(i,j) = I_{MSCN}(i,j)\cdot I_{MSCN}(i+1,j) \tag{6.5}$$

Now f_2 is defined as the variance of all positive I_{VProd}, and f_3 of negative elements. For f_4 and f_5 we use μ and σ defined in equation (6.3) again. The local standard deviation σ is itself a natural image statistic useful for determining haze, and so is the coefficient of variation, so we define

$$\begin{aligned} \mu(i,j) &= \sum_{k=-K}^{K}\sum_{l=-L}^{L} w_{k,l} I_{gray}(i+k, j+l) \\ f_4 = \sigma(i,j) &= \sqrt{\sum_{k=-K}^{K}\sum_{l=-L}^{L} w_{k,l}\left(I_{gray}(i+k, j+l) - \mu(i,j)\right)^2} \\ f_5 &= \frac{\sigma(i,j)}{\mu(i,j)} \end{aligned} \tag{6.6}$$

Contrast energy, used to define features f_6, f_7, and f_8, is a measure of local contrast. Since haze degradation heavily influences contrast, it is a useful statistic in determining the level of haze influence. The contrast is calculated using second derivatives of the Gaussian filter, thresholded to suppress noise.

$$\begin{aligned} CE(I_c) &= \frac{\alpha Z(I_c)}{Z(I_c)+\alpha k} - t_c \\ Z(I_c) &= \sqrt{(I_c * h_h)^2 + (I_c * h_v)^2} \end{aligned} \tag{6.7}$$

Here h_h and h_v are the second derivatives of Gaussian distribution in horizontal and vertical directions, α is the maximal value that I_c can take, k is contrast gain, t_c is noise suppression

threshold and $c \in \{gray, rg, yb\}$. Gaussians are fixed at standard deviation 0.12 and mean 0, k is 0.1 for all channels and t_c is 0.2353, 0.0528 and 0.2287 for gray, rg and yb respectively. The channels are defined as

$$\begin{aligned} gray &= 0.299R + 0.587G + 0.114B \\ rg &= R - G \\ yb &= \frac{R+G}{2} - B \end{aligned} \tag{6.8}$$

Since haze degrades some information, it is reasonable to expect that a hazy image contains less information than it would if it were not hazy. In order to use this to measure the level of dehazing, we use the entropy of the image defined as

$$IE(I) = -\sum p(h_i) \log[p(h_i)] \tag{6.9}$$

Since fog reduces saturation, it is also used as a statistic. Saturation is calculated using the standard HSV color space definition

$$S(i,j) = \frac{\max\limits_{c \in \{R,G,B\}} I_c(i,j) - \min\limits_{c \in \{R,G,B\}} I_c(i,j)}{\max\limits_{c \in \{R,G,B\}} I_c(i,j)} \tag{6.10}$$

Colorfulness is calculated as

$$f_{12} = \sqrt{\sigma_{rg}^2 + \sigma_{yb}^2} + 0.3\sqrt{\mu_{rg}^2 + \mu_{yb}} \tag{6.11}$$

Once again rg and yb are defined as in equation (6.8) and σ and μ denote standard deviation and mean.

The features that are calculated per pixel, namely f_4, f_5, f_6, f_7, f_8, f_{10} and f_{11} , are then averaged over the entire patch in order to get a 12-dimensional vector for each patch. The set of vectors is then used to estimate the parameters of a 12-dimensional Gaussian distribution v_2 and Σ_2 using standard maximum likelihood estimation techniques. Then we measure the distance of the estimated distribution from a precomputed distribution from a large number of foggy images using the Mahalanobis distance

$$D_f(v_1, v_2, \Sigma_1, \Sigma_2) = \sqrt{(v_1 - v_2)^T \left(\frac{\Sigma_1 + \Sigma_2}{2}\right)^{-1} (v_1 - v_2)} \tag{6.12}$$

Here v_1 and Σ_1 are parameters of the distribution that was learned over a large number of haze-free images. A second distance D_{ff} is then calculated using the same method but with parameters estimated from a large number of hazy images. the final FADE metric is then calculated

as

$$D = \frac{D_f}{D_{ff} + 1} \tag{6.13}$$

The addition of 1 in the denominator is used for stabilization to avoid issues of numerical stability with small denominators. A smaller number indicates smaller D_f, and so natural scene statistics more like those of haze-free images, indicating better dehazing.

We evaluated the FADE metric on the dataset included in the paper that proposed it for all of the methods mentioned above. The results are shown in table 6.6. Here our methods outperform the state-of-the-art supervised methods. Examples of an image from the dataset and dehazed versions of the image can be seen in Figure 6.3. The supervised methods barely change the image, removing very little haze, and because of this, improving the FADE score very little.

Model	FADE score
Baseline	1.7482
VarDCP	0.7128
UnsupervisedDCP	**0.529**
FFANet	1.9541
Dehazeformer	2.076

Table 6.6: FADE scores of several different methods on the dataset introduced in the FADE paper. A lower score indicates better dehazing. Proposed models are written in bold letters. Baseline refers to FADE scores of images with no dehazing applied.

Figure 6.3: Top left is the original image, top middle image dehazed using VarDCP, and top right using Unsupervised DCP. Bottom left is dehazed using FFANet, bottom middle using Dehazeformer, and bottom right using DCP. FFANet and Dehazeformer barely affect the image at all. The DCP removes most of the haze, but the resulting contrast of the colors is a lot too high and look slightly unnatural. The same is true to a lesser extent for the VarDCP method. The Unsupervised DCP method produces the most natural-looking image in this case.

6.1.4 Task driven

For task-driven comparison, we used the RTTS subset of the RESIDE dataset and the DAWN dataset. We used a pre-trained Faster-RCNN for all detections and compared the detection results to the baseline of not doing any dehazing. Table of mAR scores on RTTS is shown in Table 6.7, and scores on the DAWN dataset are shown in Table 6.8. Note that we are using mean average recall rather than mean average precision, as discussed in section 3. Examples of detection in hazy and dehazed images can be seen in figure 6.4. The dehazed image is processed using the Unsupervised DCP method. In the dehazed image, we can see additional detections of both cars, on the right side of the image, and cyclists, on the left side of the image. Only detections with a score of 0.8 or above are shown.

For many practical purposes, this task-driven test is the most important one. This shows how well methods can be used for preprocessing to improve the quality of other algorithms. We can see that both FFANet and Dehazeformer reduce the rate of detection for pedestrians in the DAWN dataset. This validates our assumption that, while supervised methods have good results on synthetic datasets, they do not generalize well to real hazy images.

Model	Pedestrian mAR	Car mAR
Baseline	42.87	32.60
VarDCP	**44.38**	33.98
UnsupervisedDCP	44.04	34.22
FFANet	43.59	**34.37**
Dehazeformer	43.85	34.31

Table 6.7: Mean average recall on RTTS dataset. Higher numbers mean better detection. Proposed methods are written in bold letters. Baseline refers to mean average recall of images with no dehazing applied.

Model	Pedestrian mAR	Car mAR
Baseline	53.09	40.17
VarDCP	**54.01**	40.19
UnsupervisedDCP	53.59	40.22
FFANet	46.42	38.90
Dehazeformer	52.47	**40.39**

Table 6.8: Mean average recall on DAWN dataset. Higher numbers mean better detection. Proposed methods are written in bold letters. Baseline refers to detection scores of images with no dehazing applied. Note that both of our methods improved the detection rate, at least slightly, for both cars and pedestrians, while the methods based on supervised learning reduced the rate of detection of pedestrians.

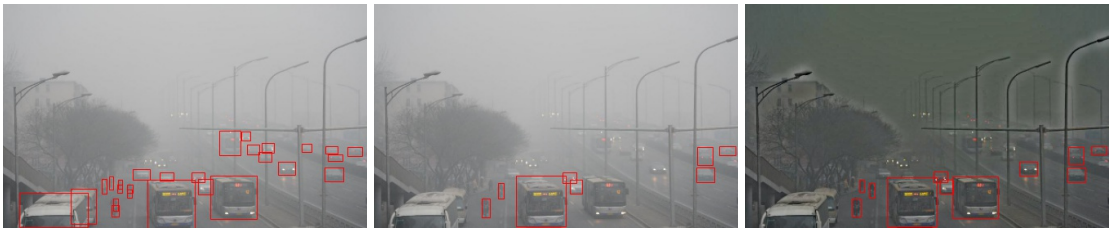

Figure 6.4: Detections of cars and pedestrians on the DAWN dataset. Left is the ground truth image with all of the cars and pedestrians annotated by hand. The middle is the detections done on the hazy image. Right is the detections done on the dehazed image. The dehazing is done using the Unsupervised DCP method. Even though the dehazing is not perfect in this case, there are additional detections in both cars, on the right side of the image, and cyclists, on the left side of the image.

6.1.5 Visual inspection

Visual inspection was done on a set of natural hazy images. These images are of different urban and rural scenes and are commonly used to evaluate single image dehazing methods in many papers. Results can be seen in Figure 6.5 and Figure 6.6. Note how, on both of these images, the transformer and FFA methods barely change the hazy image, while both of our proposed methods restore a large part of the saturation. Also, in the hay cone image, we can see that the top left corner of the variational method suffers from oversaturation, this is probably due to the assumption that A is constant that is made in that method, resulting in slight errors in estimating A in those pixels. Since t is very small in those pixels, this causes large overall errors. The road image also shows a much larger improvement using our methods over the state-of-the-art ones. The comparison between the variational method and the unsupervised one is also interesting, as the variational method increases contrast much more, but the colors may even seem too saturated to be natural in some places. Interestingly, the Multi-Scale Convolutional Neural Network [49] also shows relatively good visual results. This is also a supervised method but has much fewer parameters than Restormer and FFANet, so it cannot overfit quite as much as they do onto the haze degradation model.

Figure 6.5: Top left is the original image, top middle image dehazed using VarDCP, and top right using Unsupervised DCP. Bottom left is dehazed using FFANet, bottom middle using Dehazeformer, and bottom right using MSCNN. Note that the biggest improvements seem to be from the VarDCP method and the unsupervised DCP method, while the smallest improvements seem to come from the FFANet and Dehazeformer.

The train image in Figure 6.7 is especially interesting, as it contains the train headlights. These direct light sources are problematic for dehazing since they do not follow the model given in equation (1.20). This can most easily be seen by the presence of some diffusion of light in the haze around the headlights, which can not be accounted for by the model. The brightness of the lights can also cause problems for DCP-based airlight estimation. The unsupervised method

Figure 6.6: Top left is the original image, top middle image dehazed using VarDCP, and top right using Unsupervised DCP. Bottom left is dehazed using FFANet, bottom middle using Dehazeformer, and bottom right using MSCNN. Note that the biggest improvements seem to be from the VarDCP method and the unsupervised DCP method, while the smallest improvements seem to come from the FFANet and Dehazeformer.

does manage to remove some of the haze, but the result of the transformer and of the VarDCP are the most visually dehazed ones.

Figure 6.7: Top left is the original image, top middle image dehazed using VarDCP, and top right using Unsupervised DCP. Bottom left is dehazed using FFANet, bottom middle using Dehazeformer, and bottom right using MSCNN. The biggest improvement here comes from the Dehazeformer, followed by the VarDCP method. This image is particularly interesting because the headlights of the train do not conform to the haze degradation model well.

The image containing the mountain in Figure 6.8 is an example of the Unsupervised DCP method underperforming. The sky is too blue, and the rest of the image seems to have slightly too high saturation values as well. This can be partly explained by the large amount of non-haze-obscured sky in the image. Sky does not satisfy the dark channel prior, and clouds can interfere with airlight estimation due to their brightness. FFA Net and Dehazeformer do not seem to

remove much haze from the image, but neither method degrades the image further. Dealing with large monochrome surfaces that do not conform to the prior is definitely a weakness of the proposed methods that needs to be addressed in the future.

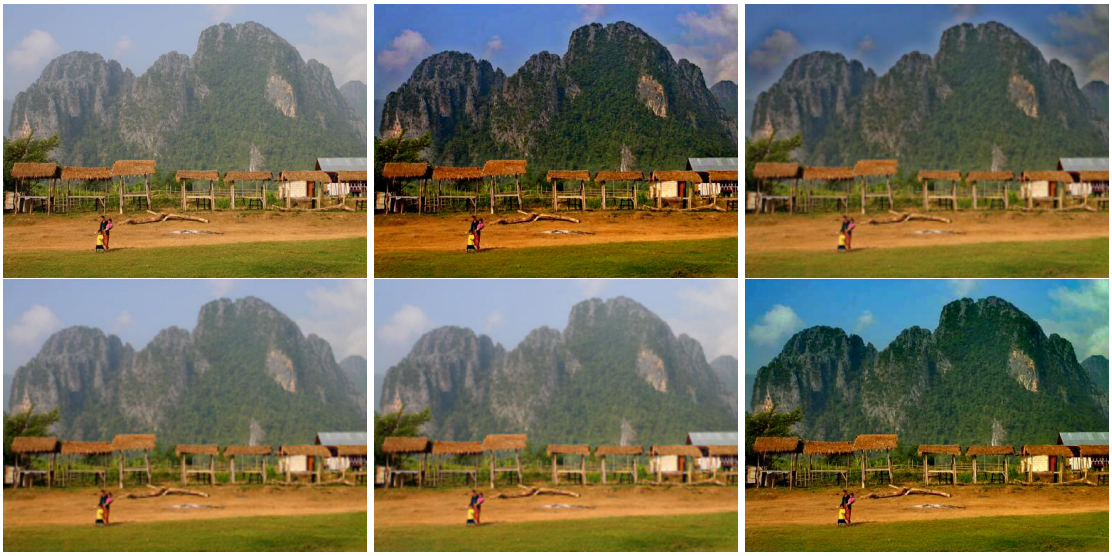

Figure 6.8: Top left is the original image, top middle image dehazed using VarDCP, and top right using Unsupervised DCP. Bottom left is dehazed using FFANet, bottom middle using Dehazeformer, and bottom right using MSCNN. The Unsupervised DCP method underperforms in this example. The FFANet and Dehazeformer still do not change the image much at all.

The aerial image of Dubai is an example of where our methods deal with the sky well. The images seem dehazed, with saturation restored without the result being oversaturated and visually displeasing. Our explanation for this is that in this image, the sky was also partially obscured by haze, and there were fewer clouds. All this combined means that the dark channel prior-based estimation of airlight worked better. The colors look the most natural in the VarDCP method, while the unsupervised method is just slightly worse. In this image, the colors of the MSCNN based reconstruction seem a bit off, probably due to a slightly incorrect estimation of the airlight.

In Figure 6.10, we see the effects of dehazing on an image taken in a forest. In this image, we can see signs of halo effects of haze around the leaves near the camera in both of our proposed methods. This is because the only regularization used on the transmission map is total variation, which ensures sharp edges in t. In the standard DPC method, this is dealt with using the guided image filtering [4], and in supervised learning methods the networks learn that the edges of t should coincide with edges in observed image I. This area of our variational and unsupervised methods still needs work, perhaps by including the prior from the variational method for enhanced visualization introduced in [12]. The branches and leaves in the MSCNN method are slightly oversaturated, while Dehazeformer and FFANet did not remove practically any haze from the image.

In Table 6.9, we can see the results of the previously introduced FADE metric on dehazed

Figure 6.9: Top left is the original image, top middle image dehazed using VarDCP, and top right using Unsupervised DCP. Bottom left is dehazed using FFANet, bottom middle using Dehazeformer, and bottom right using MSCNN. FFANet and Dehazeformer once again do not change the image. The best results come from VarDCP and the Unsupervised DCP.

Figure 6.10: Top left is the original image, top middle image dehazed using VarDCP, and top right using Unsupervised DCP. Bottom left is dehazed using FFANet, bottom middle using Dehazeformer, and bottom right using MSCNN. This example is interesting as it shows a need for additional processing or additional regularization of the transmission map in our methods to avoid the halo effect.

images. Our two proposed methods achieve better results than the supervised learning-based methods on all of the images, except for the mountain one, where MSCNN outperforms the unsupervised method, and the train one, where the Dehazeformer outperforms it. This is useful as it shows that the results of the FADE metric are the same as those of our visual inspection, validating the metric. It is also important to note that the MSCNN method outperforms the other two supervised learning methods, even though it has significantly worse results on full reference metrics on synthetic datasets. This is further proof that FFANet and Dehazeformer are severely

overfitting on the synthetically generated images and so lose the ability to generalize properly.

	Original	VarDCP	UDCP	MSCNN	FFANet	Dehazeformer
Hay Cones	0.6805	**0.1718**	0.1747	0.3418	0.7022	1.0618
Road	1.0101	**0.2888**	0.4329	0.4459	1.1956	1.2101
Mountain	0.6211	**0.2696**	0.3261	0.2973	0.7224	0.7386
Forest	0.4235	**0.2171**	0.2868	0.2366	0.5497	0.6197
Train	1.398	**0.3027**	0.4905	0.960	0.9456	0.4548
Dubai	1.1147	0.3818	**0.3510**	0.3846	1.1387	0.9922

Table 6.9: Results of reference free FADE metric for image dehazing [45] on the images shown from this section. Smaller numbers indicate better dehazing. We can see that the proposed methods outperform the supervised learning-based methods on almost all of the images. We can also see that the scores do, in general, come to the same conclusion as our visual inspection of dehazing, validating the FADE metric.

6.2 Rain removal

In order to properly test how well our methods work, we need to compare them to state-of-the-art methods for the same task. For rain removal, we have chosen to compare our proposed method to the Unidirectional Global Sparse Method (UGSM) [26], which inspired our method. We will also compare the results to Restormer, a transformer architecture for single image rain removal, and Progressive Image Derain Network (ProgNet) [32]. ProgNet (100L) refers to the ProgNet trained on the Rain100L dataset, and ProgNet (14k) refers to the same network trained on the Rain14k dataset.

6.2.1 Choosing the weight matrix

A good choice of the weight matrix is obviously important to both of the proposed rain removal techniques. As a proof of concept, in order to properly test if a good choice of W can make a difference, we first did tests using ground truth information for constructing W. We took the observed image S, subtracted the ground truth image J, and took the resulting image R as a ground truth map of rain. We then mapped that using a thresholding function

$$W(i,j) = \begin{cases} 1 & R(i,j) > t \\ 0 & R(i,j) \leq t \end{cases} \tag{6.14}$$

In our experiments, we used the threshold 0.2. We tested W defined like this on several images and compared it to $W \equiv 1$. This proof of concept only served to show that a proper choice of

W can improve the results, and it does. An example of this can be seen in Figure 6.11. This showed that a good choice of W can both help remove more rain and reduce smoothing non-rain edges.

Figure 6.11: Top left is the ground truth rain-free image, top right is the image with added rain, bottom left is the version made with the proposed method using optimal W, and bottom right is the version derained with $W = 1$ everywhere. We can see that the rain is better removed, and other parts of the image are better preserved.

In practice, however, the ground truth information about rain is unavailable. The method proposed in [23], which we use for detecting rain, is both slow, often taking more than 10 seconds per image, and does not detect all of the rain streaks in many cases. Because of this, we decided to try and use several simple image statistics to detect potential rainy pixels. Overdetection is a smaller problem than underdetection for our method, as not detected streaks will not be removed, and detected non-streaks will only be slightly smoothed out. The first statistic that we tried out was based on saturation. Rain streaks appear as white or very light gray in most images, meaning that pixels that are more highly saturated are almost certainly not rain pixels. The rain map W is, in this case, constructed as

$$W := 1 - M + m \tag{6.15}$$

Here M denotes the largest value over all the channels in that pixel, and m is the smallest value. The term $m - M$ is 0 for gray pixels and is larger for pixels further from monochrome. The $1-$ inverts this and maps the values back into interval $[0, 1]$. A second image statistic we tried was

Figure 6.12: Comparison of minimal channel weighted deraining (bottom left) and one where weights are defined by saturation (bottom right). The top row contains the ground truth and the generated rainy image.

brightness. Pixels influenced by rain in most images appear brighter than their surroundings. To use this fact, we can define W as

$$W := \min_{c \in \{R,G,B\}} S_c = m \tag{6.16}$$

If at least one of the channels in a pixel is dark, then that pixel is probably not influenced by rain, so the value of W will be low. These two were the most successful methods we created, the one based on the value in the minimal channel being more successful. A comparison of these two methods can be seen in Figure 6.12. We have also tried averaging the results of the two methods as

$$W := \frac{W_{saturation} + W_{brightness}}{2} = \frac{1 - M + m + m}{2} \tag{6.17}$$

However, this combination underperformed compared to both methods on their own. It is also important to note that averaging multiple different choices of W brings W closer to a constant, and a constant W is equivalent to $W \equiv 1$, which we know leads to over-smoothing. A comparison of using $W \equiv 1$ and $W = m$ can be seen in Figure 6.13. Note how much better the details in the rock are preserved when using $W = m$. However, there is also the downside of not fully removing all of the rain from other parts of the image.

We also attempted to use machine learning, or even deep learning, to produce a better choice

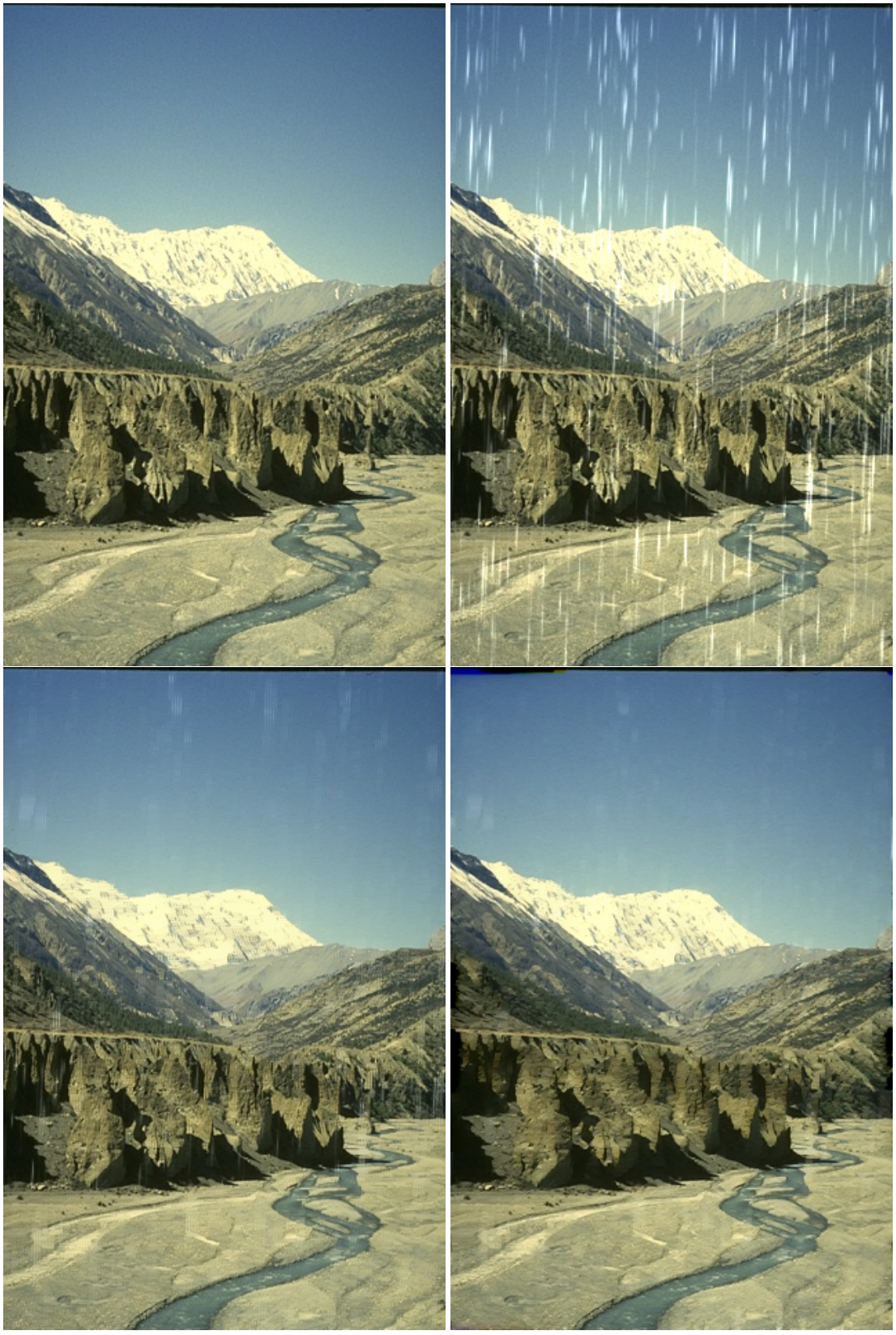

Figure 6.13: Example of deraining using $W \equiv 1$ on bottom right and using $W = m$ on bottom left. In the top row are the ground truth and the rainy version of the image. Note that there is much less smoothing on the rocks when using $W = m$.

of W. However, the only way to do that we found was to use supervised learning on synthetic datasets, which is a problem. The point of a variational or unsupervised approach is to ensure better generalization than supervised methods trained on synthetic datasets, so we wanted to avoid training parts of our method in a supervised manner.

6.2.2 Hyperparameter choices

The choice of all hyperparameters is very important. In the previous section, we focused on specific methods to choose W. However, there are still more parameters to specify. For the most part, these parameters are independent of the image, but some may benefit from being adapted to each image if it is possible to do that automatically. In table 6.10, we see the parameters used in all of the experiments when using the variational method. Values of λ_1 and λ_2 are both

Parameter	Value
λ_1	$\frac{1}{\text{mean}(W)}$
λ_2	$\frac{0.1}{\text{mean}(W)}$
β_1	200
β_2	200
β_3	200
ε	10^{-3}
n_{max}	600

Table 6.10: Table showing values of parameters used in all experiments using the variational deraining method. Note that λ_1 and λ_2 are both adaptive to the image because they incorporate information about W.

adaptive to the image, as they are dependant on the value of mean(W). This allows us not to worry about the scale of W, as it cancels out with the terms that include it in the functional. Additionally, if very little rain is detected, so W is close to 0 in most of the image, the parts that do include rain will be very smoothed. If there is almost no localization in the rain, so W is close to constant, the smoothing will be weaker. This is a good way to use varying of W to either more strongly smooth out parts that are affected by rain for sure or more weakly smooth out the entire image, and so preserve some of the natural edges.

6.2.3 Full-reference

For full-reference metrics, we again used PSNR and SSIM. We compared the performance of models on the datasets they were trained on and on datasets that were generated in a different way. This comparison to different datasets allows us to see how well models generalize and

capture the concept of rain removal and how much they overfit on a specific method of rain streak simulation.

In Table 6.11, we see the results of testing on the Rain100L synthetic dataset. Since ground truth is available for all images, we use the PSNR and SSIM metrics for evaluation. Most of the methods, other than our proposed method, are trained using supervised learning on datasets generated in similar ways to Rain100L. Even with this advantage of potentially overfitting to the model that generates rain rather than removing real rain, our proposed method still outperforms many of them in terms of SSIM. Importantly, we also outperform the original UGSM method in terms of both SSIM and PSNR, which shows that introducing weights improves the final results. In Table 6.12, we can see the results on another synthetic dataset, Rain14k. The Restormer has

Method	PSNR	SSIM
Weighted directional total variation	28.10	0.936
UGSM	27.13	0.8943
UGMS CNN [50]	29.18	0.923
DerainNet [51]	27.03	0.884
Restormer	**37.52**	**0.9712**
ProgNet (100L)	35.01	0.9675
ProgNet (14k)	26.83	0.8529

Table 6.11: PSNR and SSIM results on Rain100L dataset of several methods. Our proposed method is written in bold letters. Higher numbers indicate better performance on both metrics.

been trained on a combination of multiple datasets and so generalizes well to both synthetic datasets. However, the ProgNet has problems generalizing when trained on one and evaluated on the other. Our proposed method also underperforms in this example. We believe that this is because this way of generating rainy images is less realistic and misses the features we rely on in the proposed method. Most notably, the rain streaks are not brighter than their environment, and so our weight generation strategies work poorly.

In Table 6.13, we can see the results on the RealRain dataset. The performance of the Weighted directional total variation method here is closer to the state-of-the-art results, almost as good in terms of PSNR and even better in terms of SSIM. This again validates the idea that many of these supervised methods overfit onto the generated images rather than learning real scene statistics related to rain degradation. In Figure 6.15, we can see a visual comparison of results on this dataset.

Method	PSNR	SSIM
Baseline	23.82	0.7564
Weighted directional total variation	24.29	0.8069
Restormer	**32.04**	**0.9249**
ProgNet (100L)	25.72	0.8162
ProgNet (14k)	30.49	0.9076

Table 6.12: PSNR and SSIM results on Rain14k dataset of several methods. Our proposed method is written in bold letters. Higher numbers indicate better performance on both metrics. Baseline refers to performing no deraining at all.

Figure 6.14: An example of a generated rainy image and derained images from the Rain14k dataset. The top row contains the ground truth clear image on the left, the generated rainy image in the middle, and the image with rain removed using our proposed method on the right. The bottom row contains the image derained with ProgNet (100L) on the right, ProgNet (14k) in the middle, and Restormer on the left.

Method	PSNR	SSIM
Baseline	22.51	0.8500
Weighted directional total variation	22.46	**0.9253**
Restormer	**24.76**	0.9172
ProgNet (14k)	23.89	0.8921
ProgNet(100L)	22.67	0.8696

Table 6.13: PSNR and SSIM results on RealRain dataset of several methods. Our proposed method is written in bold letters. Higher numbers indicate better performance on both metrics.

6.2.4 Visual inspection

For visual inspection, we took one real rainy image and one image from the Real Rain dataset. In Figure 6.15, we can see the results of rain-streak removal on an image from the Real Rain dataset. The ground truth image is constructed from video using additional frames, and the rainy image is just one single frame. Note that our proposed method removed rain visually better than the state-of-the-art methods.

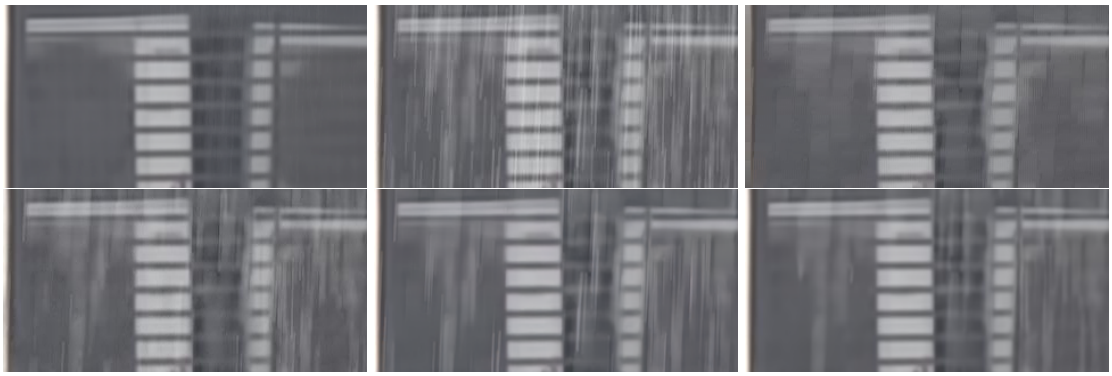

Figure 6.15: Example of an image from the real rain dataset. The top row contains the image extracted from the video on the left, a single frame of rain in the middle, and the image with rain removed using our proposed method on the right. The bottom row contains the image derained with ProgNet (100L) on the left, ProgNet (14k) in the middle, and Restormer on the right.

In Figure 6.16, we can see an example of rain-streak removal on a real rainy image. Note that here our proposed method removes by far the most rain, followed by the Restormer. Importantly, in this image of real rain, rain streaks follow most of the priors built into our method, validating our assumptions.

Figure 6.16: Example of rain removal on a real rainy image. Top left is the ground truth, top right is removal using our proposed method, bottom left is using the ProgNet, and bottom right using Restormer.

Chapter 7

Conclusion

7.1 Conclusion of the thesis

In this thesis, we developed new methods for restoring images degraded by fog or rain. The proposed methods are based on optimizing a functional made for the specific problem. For single image dehazing, we first performed a detailed error analysis to determine how the error in estimating the parameters of the haze propagates to the estimation of the dehazed image. After that, we derived a smooth approximation of the dark channel prior, one of the most successful priors used in single image dehazing. We used this smooth approximation and insights from the error analysis to create a functional that, when minimized, gives a good approximation of the dehazed image. We also developed an optimization scheme to minimize this functional efficiently. We used the same functional as a loss function for an unsupervised deep learning method. For the problem of single image rain removal, we developed a functional based on weighted directional total variation. We developed a method to minimize the functional and explored the effects the weights have on the minimizer of the functional.

We compared our proposed methods with state-of-the-art deep learning based methods. The state-of-the-art methods are based on supervised learning and use synthetically created datasets to learn from. Because of this, while they outperform our methods on synthetic data, our methods perform better on real data. For comparison of quality on real-world data, where the ground truth is unknown, we used reference-free metrics and task-driven testing. The task-driven testing was based on testing the improvement of performance of a pre-trained object detector on images dehazed using different dehazing methods. In all of these experiments on real-world data, our proposed methods outperformed the supervised learning based methods.

7.2 Direction of future research

There are many potential directions for future research. One possible direction is deriving smooth approximations of priors used in other classical methods and combining multiple priors into one method. This could be useful both in terms of variational methods and in terms of loss functions for deep learning based methods. Another important direction is the development of more reference-free metrics, which allow us to compare the quality of image restoration without access to ground truth. We see potential in linking the currently used reference-free metrics to deep learning, such as the patch distribution modeling framework used in anomaly detection.

The presented methods show that the classical prior-based approaches have a lot of merit to them, they just need to be adjusted properly to be combined with the new data-driven methods. We think that in the future, this has the potential to significantly improve the quality of image restoration methods.

Bibliography

[1]Lowe, D., “Object recognition from local scale-invariant features”, in Proceedings of the Seventh IEEE International Conference on Computer Vision, Vol. 2, 1999, str. 1150-1157 vol.2.

[2]Narasimhan, S. G., Nayar, S. K., “Vision and the atmosphere”, International Journal of Computer Vision, Vol. 48, No. 3, Jul 2002, str. 233-254, available on: https://doi.org/10.1023/A:1016328200723

[3]He, K., Sun, J., Tang, X., “Single image haze removal using dark channel prior”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 33, No. 12, 2011, str. 2341-2353.

[4]He, K., Sun, J., Tang, X., “Guided image filtering”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 35, No. 6, 2013, str. 1397-1409.

[5]Liu, F., Yang, C., “A fast method for single image dehazing using dark channel prior”, in 2014 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC), 2014, str. 483-486.

[6]Shuai, Y., Liu, R., He, W., “Image haze removal of wiener filtering based on dark channel prior”, in 2012 Eighth International Conference on Computational Intelligence and Security, 2012, str. 318-322.

[7]Jackson, J., Ariyo, O., Acheampong, K., Boakye, M., Frimpong, E., Ashalley, E., Rao, Y., “Hybrid single image dehazing with bright channel and dark channel priors”, in 2017 2nd International Conference on Image, Vision and Computing (ICIVC), 2017, str. 381-385.

[8]Berman, D., Treibitz, T., Avidan, S., “Non-local image dehazing”, in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, str. 1674-1682.

[9]Hsu, W.-Y., Chen, Y.-S., “Single image dehazing using wavelet-based haze-lines and denoising”, IEEE Access, Vol. 9, 2021, str. 104 547-104 559.

[10]Ma, L., Li, Y., Zheng, M., “Underwater image enhancement using the revised haze-lines method”, in Eighth Symposium on Novel Photoelectronic Detection Technology and Applications, Su, J., Chen, L., Chu, J., Zhu, S., Yu, Q., (ur.), Vol. 12169, International Society for Optics and Photonics. SPIE, 2022, str. 1216951, available on: https://doi.org/10.1117/12.2624280

[11]Zhu, Q., Mai, J., Shao, L., “A fast single image haze removal algorithm using color attenuation prior”, IEEE Transactions on Image Processing, Vol. 24, No. 11, 2015, str. 3522-3533.

[12]Fang, F., Wang, T., Wang, Y., Zeng, T., Zhang, G., “Variational single image dehazing for enhanced visualization”, IEEE Transactions on Multimedia, Vol. 22, No. 10, 2020, str. 2537-2550.

[13]Ulyanov, D., Vedaldi, A., Lempitsky, V., “Deep image prior”, arXiv:1711.10925, 2017.

[14]Gandelsman, Y., Shocher, A., Irani, M., ““double-dip”: Unsupervised image decomposition via coupled deep-image-priors”, in 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, str. 11 018-11 027.

[15]Cai, B., Xu, X., Jia, K., Qing, C., Tao, D., “Dehazenet: An end-to-end system for single image haze removal”, IEEE Transactions on Image Processing, Vol. 25, No. 11, 2016, str. 5187-5198.

[16]Song, Y., He, Z., Qian, H., Du, X., “Vision transformers for single image dehazing”, available on: https://arxiv.org/abs/2204.03883 2022.

[17]Qin, X., Wang, Z., Bai, Y., Xie, X., Jia, H., “Ffa-net: Feature fusion attention network for single image dehazing”, Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, No. 07, Apr. 2020, str. 11 908-11 915, available on: https://ojs.aaai.org/index.php/AAAI/article/view/6865

[18]N., B. R., N, V., “Single image haze removal using a generative adversarial network”, available on: https://arxiv.org/abs/1810.09479 2018.

[19]Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y., “Generative adversarial networks”, available on: https://arxiv.org/abs/1406.2661 2014.

[20]Zhu, J.-Y., Park, T., Isola, P., Efros, A. A., “Unpaired image-to-image translation using cycle-consistent adversarial networks”, in Computer Vision (ICCV), 2017 IEEE International Conference on, 2017.

[21]Simonyan, K., Zisserman, A., “Very deep convolutional networks for large-scale image recognition”, available on: https://arxiv.org/abs/1409.1556 2014.

[22]Li, B., Peng, X., Wang, Z., Xu, J., Feng, D., “Aod-net: All-in-one dehazing network”, in 2017 IEEE International Conference on Computer Vision (ICCV), 2017, str. 4780-4788.

[23]Wang, Y., Liu, S., Chen, C., Xie, D., Zeng, B., “Rain removal by image quasi-sparsity priors”, available on: https://arxiv.org/abs/1812.08348 2018.

[24]Chang, Y., Yan, L., Zhong, S., “Transformed low-rank model for line pattern noise removal”, in 2017 IEEE International Conference on Computer Vision (ICCV), 2017, str. 1735-1743.

[25]Chen, Y.-L., Hsu, C.-T., “A generalized low-rank appearance model for spatio-temporally correlated rain streaks”, in 2013 IEEE International Conference on Computer Vision, 2013, str. 1968-1975.

[26]Deng, L.-J., Huang, T.-Z., Zhao, X.-L., Jiang, T.-X., “A directional global sparse model for single image rain removal”, Applied Mathematical Modelling, Vol. 59, 2018, str. 662-679, available on: https://www.sciencedirect.com/science/article/pii/S0307904X18301069

[27]Chen, H., Xu, Z., Zhang, Y., Fan, Y., Li, Z., “An l0-regularized global anisotropic gradient prior for single-image de-raining”, Applied Mathematical Modelling, Vol. 98, 2021, str. 628-651, available on: https://www.sciencedirect.com/science/article/pii/S0307904X2100202X

[28]Fu, Y.-H., Kang, L.-W., Lin, C.-W., Hsu, C.-T., “Single-frame-based rain removal via image decomposition”, in 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2011, str. 1453-1456.

[29]Kang, L.-W., Lin, C.-W., Fu, Y.-H., “Automatic single-image-based rain streaks removal via image decomposition”, IEEE Transactions on Image Processing, Vol. 21, No. 4, 2012, str. 1742-1755.

[30]Aharon, M., Elad, M., Bruckstein, A., “K-svd: An algorithm for designing overcomplete dictionaries for sparse representation”, IEEE Transactions on Signal Processing, Vol. 54, No. 11, 2006, str. 4311-4322.

[31]Wang, Y., Liu, S., Chen, C., Zeng, B., “A hierarchical approach for rain or snow removing in a single color image”, IEEE Transactions on Image Processing, Vol. 26, No. 8, 2017, str. 3936-3950.

[32]Ren, D., Zuo, W., Hu, Q., Zhu, P., Meng, D., “Progressive image deraining networks: A better and simpler baseline”, in IEEE Conference on Computer Vision and Pattern Recognition, 2019.

[33]Zamir, S. W., Arora, A., Khan, S., Hayat, M., Khan, F. S., Yang, M.-H., “Restormer: Efficient transformer for high-resolution image restoration”, in CVPR, 2022.

[34]Park, Y., Jeon, M., Lee, J., Kang, M., “Mcw-net: Single image deraining with multi-level connections and wide regional non-local blocks”, Signal Processing: Image Communication, Vol. 105, 2022, str. 116701, available on: https://www.sciencedirect.com/science/article/pii/S0923596522000431

[35]Du, Y., Xu, J., Qiu, Q., Zhen, X., Zhang, L., “Variational image deraining”, in 2020 IEEE Winter Conference on Applications of Computer Vision (WACV), 2020, str. 2395-2404.

[36]Li, B., Ren, W., Fu, D., Tao, D., Feng, D., Zeng, W., Wang, Z., “Benchmarking single-image dehazing and beyond”, IEEE Transactions on Image Processing, Vol. 28, No. 1, 2019, str. 492–505.

[37]Silberman, N., Hoiem, D., Kohli, P., Fergus, R., “Indoor segmentation and support inference from rgbd images”, in Computer Vision – ECCV 2012, Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C., (ur.). Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, str. 746–760.

[38]Scharstein, D., Szeliski, R., “High-accuracy stereo depth maps using structured light”, in 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings., Vol. 1, 2003, str. I-I.

[39]Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B., “The cityscapes dataset for semantic urban scene understanding”, in Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[40]KENK, M., “Dawn”, available on: https://data.mendeley.com/datasets/766ygrbt8y/3 2020.

[41]Ancuti, C. O., Ancuti, C., Sbert, M., Timofte, R., “Dense haze: A benchmark for image dehazing with dense-haze and haze-free images”, in IEEE International Conference on Image Processing (ICIP), ser. IEEE ICIP 2019, 2019.

[42]Ancuti, C. O., Ancuti, C., Timofte, R., Vleeschouwer, C. D., “I-haze: a dehazing benchmark with real hazy and haze-free indoor images”, in arXiv:1804.05091v1, 2018.

[43]Ancuti, C. O., Ancuti, C., Timofte, R., Vleeschouwer, C. D., “O-haze: a dehazing benchmark with real hazy and haze-free outdoor images”, in IEEE Conference on Computer Vision and Pattern Recognition, NTIRE Workshop, ser. NTIRE CVPR’18, 2018.

[44]Ancuti, C. O., Ancuti, C., Timofte, R., Gool, L. V., Zhang, L., Yang, M.-H., “Ntire 2019 image dehazing challenge report”, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, ser. IEEE CVPR 2019, 2019.

[45]Choi, L. K., You, J., Bovik, A. C., “Referenceless prediction of perceptual fog density and perceptual image defogging”, IEEE Transactions on Image Processing, Vol. 24, No. 11, 2015, str. 3888-3901.

[46]Yang, W., Tan, R. T., Feng, J., Liu, J., Guo, Z., Yan, S., “Deep joint rain detection and removal from a single image”, in 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, str. 1685-1694.

[47]Fu, X., Huang, J., Zeng, D., Huang, Y., Ding, X., Paisley, J., “Removing rain from single images via a deep detail network”, in 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, str. 1715-1723.

[48]Wang, T., Yang, X., Xu, K., Chen, S., Zhang, Q., Lau, R. W., “Spatial attentive single-image deraining with a high quality real rain dataset”, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2019.

[49]Ren, W., Liu, S., Zhang, H., Pan, J., Cao, X., Yang, M.-H., “Single image dehazing via multi-scale convolutional neural networks”, in Computer Vision – ECCV 2016, Leibe, B., Matas, J., Sebe, N., Welling, M., (ur.). Cham: Springer International Publishing, 2016, str. 154–169.

[50]Yasarla, R., Patel, V. M., “Uncertainty guided multi-scale residual learning-using a cycle spinning cnn for single image de-raining”, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2019, str. 8405-8414.

[51]Fu, X., Huang, J., Ding, X., Liao, Y., Paisley, J., “Clearing the skies: A deep network architecture for single-image rain removal”, IEEE Transactions on Image Processing, Vol. 26, No. 6, 2017, str. 2944-2956.

List of Figures

List of Tables

Biography

Vedran Stipetić was born in Zagreb in 1994. where he attended his primary school and secondary school, V Gymnasium. He received his bachelor's degree in mathematics from the Faculty of Natural Sciences and Mathematics at the University of Zagreb and his master's degree in mathematical modeling and computation from the Technical University of Denmark.

During his bachelor's studies, he was awarded the Rector award, and during his master's studies, he was enrolled in the honors program. Since 2019 he has been employed at the Faculty of Electrical Engineering and Computing in the Department of Electrical Systems and Information Processing, where he also started his Ph.D. His research interests include variational methods, inverse problems, and computer vision.

During his Ph.D. studies, he was involved in organizing multiple summer schools, workshops, and conferences. He also taught courses Mathematical analysis 1 and Mathematical analysis 2.

He participated in the work of the state committee for mathematical competitions and the committee for the Croatian Mathematical Olympiad, as well as many other projects related to the popularization of science and scientific education. He is an active member and secretary of non-profit organization Udruga "Penkala".

List of publications

Journal publications

1.Stipeti ć, V., Lončarić, S., Variational formulation of dark channel prior for single image dehazing, Journal of Mathematical Imaging and Vision, Vol. 64, Issue 8, October 2022, pp. 845 - 854.

Conference publications

1.Stipeti ć, V., Lončarić, S., Local Gray World Method for Single Image Dehazing, 8th International Conference on Signal, Image Processing and Pattern Recognition (SIPP 2020), March 2020, pp. 13 - 20.

2.Stipeti ć, V., Lončarić, S., Unsupervised image dehazing using smooth approximation of dark channel prior, 7th International Conference on Frontiers of Singal Processing (ICFSP 2022), September 2022.

Conference publications not related to thesis topic

1.Stipeti ć, V., Lončarić, S., The p Curve Method for Illumination Estimation, 4th International Conference on Vision, Image and Signal Processing, December 2020.

2.Koš čević, K., Stipetić, V., Provenzi, E., Banić, N., Subašić, M., Lončarić, S., HD-RACE: Spray-based Local Tone Mapping Operator, 12th International Symposium on Image and Signal Processing and Analysis (ISPA), September 2021.

Životopis

Vedran Stipetić rođen je 1994. godine u Zagrebu. Tamo je pohađao i osnovnu školu i srednju školu, V Gimnaziju. Preddiplomski studij iz matematike završio je na Prirodoslovno-matematičkom fakultetu 2016. godine, a diplomski studij završio je na Tehničkom sveučilištu u Danskoj 2018. godine iz područja matematičkog modeliranja i računanja.

Tijekom preddiplomskog studija dobio je Rektorovu nagradu za znanstveni rad, a tijekom diplomskog studija sudjelovao je u programu izvrsnosti. Od 2019. zaposlen je na Fakultetu elektrotehnike i računarstva na Zavodu za elektroničke sustave i obradu informacija kao asistent, gdje je upisao i doktorski studij. Glavni istraživački interesi su mu varijacijske metode, inverzni problemi i računalni vid.

Tijekom doktorskog studija sudjelovao je u organizaciji mnogobrojnih konferencija, radionica i ljetnih škola. Predaje na predmetima Matematička analiza 1 i Matematička analiza 2.

Sudjelovao je u radu državnog povjerenstva iz matematike, povjerenstva za Hrvatsku matematičku olimpijadu i brojnim drugim projektima vezanima za popularizaciju znanosti. Aktivni je član i tajnik Udruge "Penkala".