Diploma Thesis

Real-Time Image Dehazing

Diplomand:	Christian T. Thurow
Fachbereich:	Computer Graphics
Matrikel-Nummer:	305341
Betreuer:	Prof. Dr.-Ing. Marc Alexa
	Dipl.-Ing. Detlef Schulz-Rückert
Abgabedatum:	19. April 2011

Erklärung

Die selbständige und eigenhändige Anfertigung versichert an Eides statt

Berlin, den April 9, 2011

Christian Thurow

Darüber hinaus versichere ich, dass ich keine anderen als die angegebenen Quellen und Hilfsmittel benutzt, sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

Acknowledgements

At this point I would like to thank the people that helped me producing this thesis. First, I thank Professor Marc Alexa(TU Berlin) and Mr. Detlef Schulz-Rückert(DFS GmbH) for giving me the opportunity to write this thesis and supporting me along the way. Next, I would like to say thanks to my proof readers, first and foremost Jan Sablatnig(TU Berlin) as well as Paul Burzlaff(University Potsdam), and Nina L. Meyer(University Würzburg). Special thanks go to my fellow student Christopher Leiste(TU Berlin) who helped me debugging the dehazing program. Last but not least, I thank Searidge Technologies for providing me with the helpful tool "CamSim" and plenty of video footage to test the dehazing program.

Abstrakt

Unsere Atmosphäre beinhaltet große Mengen an Partikeln, welche alle Bildaufnahmen im Freien beeinträchtigen. Sie lassen Szenen neblig oder diesig erscheinen, was zur Folge hat, dass die Sichtbarkeit und der Kontrast von Objekten herabgesetzt wird und beeinflusst wie leicht oder schwer wir Objekte in Bildern erkennen können. Kürzlich haben jedoch Entwicklungen in der Computer Vision ergeben, dass die Verbesserung von Außenbildaufnahmen möglich ist und Dunstschleier entfernt werden können. Viele Computer Vision Anwendungen können von dunstfreien Bildern profitieren. Diese Techniken sind physikalisch fundiert und basieren auf Theorien aus der Meteorologie und anderen Disziplinen. Leider jedoch sind diese Techniken sehr kostenintensiv, wenn es um die Komplexität geht und sind daher nicht für Echtzeit-Anwendungen geeignet. In dieser Arbeit wird eine Methode vorgeschlagen welche in der Lage ist den Dunstschleier von Bildern in Echtzeit zu entfernen, wobei ein Algorithmus mit einer Komplexität von nur $\mathcal{O}(n)$ angesetzt wird. Zum Vergleich, die meisten anderen existierenden Algorithmen auf diesem Gebiet haben eine Komplexität von $\mathcal{O}(n^2)$. Dieser Algorithmus basiert auf einer starken statistischen Annahme, dem Dark Channel Prior. Für die Evaluation wird eine neue Messmethode eingeführt wird. Diese Arbeit zeigt, dass Echtzeit-Dunstschleier-Entfernung möglich ist. Die Validität und Leistungsfähigkeit der Methode wird unter zu Hilfenahme von Flughafen Boden Überwachungs Videos evaluiert. Diese Ergebnisse erlauben es neuen Anwendungen wie hochauflösende, hochfrequente Außen-Überwachung, An-Bord-Kamera-Applikationen auf Fahrzeugen und vielen anderen, Echtzeit-Dunstschleier-Entfernung zu benutzen.

Keywords-dehaze, Dunstschleier Entfernung; dark channel prior; dehazing range quotient; real-time;

Abstract

Our atmosphere contains huge quantities of particles, compromising outdoor photography. They cause scenes to appear hazy or foggy, this reduces visibility of objects and their contrast, and makes detection of objects within the scene more difficult. However, due to recent developments in computer vision, it is now possible to improve outdoor images and remove the haze layer. Many computer vision applications can benefit from haze free images. These techniques are physically sound and based on theories from meteorology and other disciplines. Unfortunately, these techniques are so costly in terms of complexity that they are not suitable for real-time applications. In this thesis a method is proposed that is able to dehaze images in real-time, employing an algorithm with complexity of only $\mathcal{O}(n)$, whereas most existing algorithms inherit a complexity of $\mathcal{O}(n^2)$. This algorithm is based on a strong, statistically based prior, the dark channel prior. For its evaluation, a new measure for the degree of dehazing is introduced. This thesis shows that real-time dehazing is possible. It's validity and effectiveness is evaluated with video material from airport ground surveillance. These results enable real-time image dehazing for new applications like high resolution, high frame rate outdoor surveillance, on board vehicle camera applications and many more.

Keywords-dehaze; dark channel prior; dehazing range quotient; real-time;

Contents

List of Figures

List of Tables

1. Introduction

Image dehazing is a highly interdisciplinary challenge, involving meteorology, optical physics as well as computer vision and computer graphics. Haze along with fog and clouds are limiting factors for visual range in the atmosphere and heavily reduce contrast in scenes. E. Namer and Y. Y. Schechner, two of the current experts in this field described the need for dehazing algorithms as "There is a growing interest in the automatic analysis of images acquired in scattering media[..]. A main objective in such analysis is improvement of visibility and recovery of colors, as if imaging is done in clear conditions. Computer vision and human vision can then capitalize on such improved images for various applications, such as long range surveillance[..]." [Namer and Schechner, 2005]. Also, "In general, the haze-free image is more visually pleasuring. Second, most computer vision algorithms, from low-level image analysis to high-level object recognition, usually assume that the input image (after radiometric calibration) is the scene radiance. The performance of computer vision algorithms(*e.g.*, feature detection, filtering, and photometric analysis) will inevitably suffer from the biased, low-contrast scene radiance. Last, the haze removal can produce depth information and benefit many vision algorithms and advanced image editing. Haze or fog can be a useful depth clue for scene understanding. The bad haze image can be put to good use." [He et al., 2010a].

The need for image enhancement stems from the fact that the atmosphere is never free of particles. With just pure air, the visual range has been found to be between 277km [Hulbert, 1941] to 348km [Middleton, 1952], not considering the curvature of the earths surface. However, real visual ranges are much less than this theoretical value. The international visibility code for meteorological range rates visibilities between under 50m up to over 50km for exceptionally clear air. These codes have been found to reflect a convenient scale for visual ranges in the daily work of meteorologists. For the exact codes please refer to table 1.1, the scattering coefficient β_{sc} is an important parameter in visual range and will be dealt with in a separate section in this thesis(chapter 2).

Table 1.1.: International Visibility Code with Meteorological Range

Code no.	Weather Condition	Meteorological Range, R_m	Scattering coefficient, $\beta_{sc}(km^{-1})$
0	Dense fog	$\leq$ 50m	>78.2
1	Thick fog	50m - 200m	78.2 - 19.6
2	Moderate fog	200m - 500m	19.6 - 7.82
3	Light fog	500m - 1000m	7.82 - 3.91
4	Thin fog	1km - 2km	3.91 - 1.96
5	Haze	2km - 4km	1.96 - 0.954
6	Light haze	4km - 10km	0.954 - 0.391
7	Clear	10km - 20km	0.391 - 0.196
8	Very clear	20km - 50km	0.196 - 0.078
9	Exceptionally clear	>50km	0.078
-	Pure air	277km	0.0141

Source: [Hulbert, 1941]

Optical effects in the atmosphere have been subject to studies for many centuries. Early motivations have been to understand the colour shifts of distant objects and all optical effects that can be seen by observing nature through the atmosphere like the blueness of the sky or the red of the dawning sun. Also, the subject "visibility" is not a very new one, it was the time of the experimentalist H.B. de Saussure([de Saussure, 1789]) as well as Bouguer([Bouguer, 1760]) that scientists interest was drawn to "measuring the transparency of the air" for the first time. The first work, that is still to be deemed as outstanding today, is the work of Lord Rayleigh who examined light scattering in 1871 and in the following years(e.g. [Rayleigh, 1871]). With the development of airplanes a new motivation was incepted and research for "optics of the atmosphere" began to increase. At the beginning of the 19th century, scientists like Mie and Koschmieder developed theories that are still used in todays work in meteorology and computer vision. Mie(1908) expanded the theory of Rayleigh by examining larger particles such as haze aerosols, that have a great influence in scattering and extinction of light in the atmosphere. Koschmieder(1924) took the essential step from the scattering theories and developed an holistic theory for horizontal visual ranges through the atmosphere, which describes the visibility of a distant object, based on illumination, several environmental factors and the composition of the air in the scene of interest. Much of the research was driven by the first and second world war. Visibility range in various altitudes of the atmosphere, as well as the development of chemical warfare methods that were intended to reduce visibility for hostile airplane pilots were the subjects of meteorological research

at that time. Nevertheless, the work from that time is the basis for what we know about optics of the atmosphere today.

With the help of these theories, one can explain the effects that haze has on the visibility of a scene and eventually of an image taken of that scene. Moreover, with this knowledge one can even improve visibility for the human eye and develop techniques to remove haze. It is possible to improve the visibility in terms of range, colour verisimilitude and feature separation in digital images. Herein the term "dehazing" means to produce an image of a scene that does not contain haze effects although the source of that image originally comprised haze, for an example please refer to figure 1.1.

(a) Input Image (b) Output Image

Figure 1.1.: Images show the hazy input image and the dehazed result using the algorithm of [Fattal, 2008]

"Defogging" will also be subject to this work since the effects of haze and fog are to some degree similar and the transition between the two phenomena is gradual. Defogging and dehazing can greatly improve the visibility for the human eye and allow the observer to get a much higher situational awareness. Such visibility improvements may lead to a smoother and faster work flow in areas where operators need to observe a wide area under every possible weather conditions at day and night. Such as an air traffic controller at an airfield, possibly at a tower or a remote position, respectively. In recent years much work was done on dehazing algorithms utilising different kinds of computer hardware. However, as of writing this thesis, the author is not aware of any scientific work stating that dehazing was done in real-time using just the co-processing unit of a computer. However, during writing this thesis, a paper was published that describes

real-time image dehazing performed on a graphics processing unit. The main goal of this thesis is to develop a dehazing method that is capable of doing so in a real-time manner and just using the CPU of a computer. By real-time it is meant not just the in computer science often used term for predictability of processing time, but furthermore the ability of an algorithm to perform the task fast enough that the time difference between the input signal and the output is either not noticable for men or meets the requirements. The requirements in air traffic ground surveillance for example is in europe defined by the Eurocontrol as at least one update per second. The real-time ability of the algorithm will be measured on a defined test machine, as described in chapter 5, page 99. In order to achieve this goal, one must first understand the basic theories of optics, atmosphere and the visibility in the atmosphere. It is to thank authors like McCartney([McCartney, 1976]) and Middleton([Middleton, 1952]) who investigated the vast amount of literature that has been published for the subject of "optics through the atmosphere", that this subject is now feasible for workers of other scientific disciplines, by summarising the information with many references to further books and papers. This thesis will give the reader the necessary essentials of these theories and will follow the structure that is described next.

The first chapter after the introduction concerns with the theoretical base of vision through the atmosphere, followed by a chapter, which gives an overview of the dehazing methods that have been developed in recent years. In the fourth chapter one finds the work flow description of the real-time image dehazing method developed for this thesis. Next, the author evaluates the method in the second to last chapter with example video data from airports. The thesis finishes with a discussion and conclusion chapter.

2. Theoretical Considerations of Vision Through the Atmosphere

"From earliest times our principal knowledge of the physical world has been derived from the basic act of seeing" [McCartney, 1976].

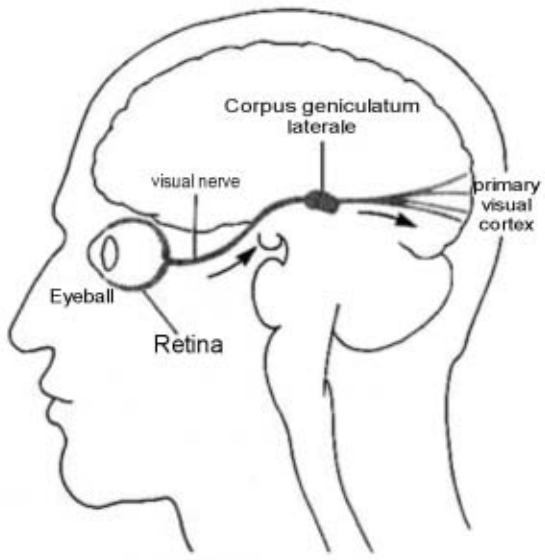
So put it McCartney in his book "Optics of the Atmosphere" in 1976. Fact is, that often the only information we have of a distant object is the visual information which traveled through and was altered by the atmosphere. We heavily rely on that information, because we use it to detect, recognise, identify and position objects. No matter whether binoculars are used or cameras that capture the image and display it on a monitor, in the end we always use our eyes. That is why this chapter will first give a brief overview of the characteristics of the human eye. In the next section, it will be concerned with the basics of the atmosphere in terms of particle and gaseous composition, followed by an extensive review of light scattering as described by Mie and Rayleigh. The very important last section of this chapter deals with the "Theorie der Horizontalen Sichtweite" of Koschmieder.

2.1. Characteristics of the Human Eye

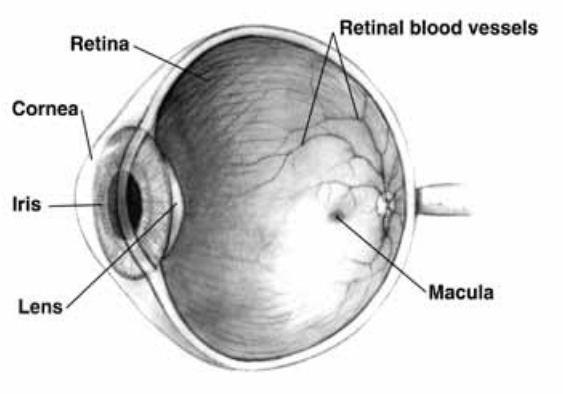
Much can be learned about the human eye from its structure. That is why this section will first describe the basic anatomy of the eye before it will discuss the consequences to seeing under various environmental conditions.

2.1.1. Anatomy of the Human Eye

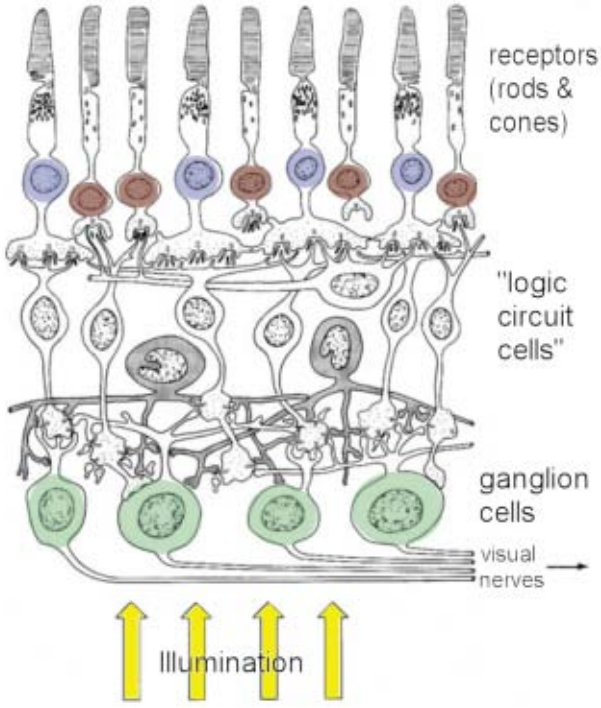
Describing every single detail of the human eye would easily exceed the scope of this thesis, hence the author will describe only the structural features that have a direct influence on the subject. The physical structure can be seen in figure 2.1, the eye itself in a medical sense is composed of the dioptric apparatus and the retina, see figure 2.1b. The dioptric apparatus is responsible for capturing light and focusing the eye to a specific distance. To mention just a few structural features, the dioptric apparatus consists of the cornea, pupil and the lens. Despite its undoubtedly fascinating capabilities of the entire eye, the focus here lies more on the retina. The retina contains the photo receptors, namely the two types cones and rods. Positioned right behind the photo receptors lies the first processing layer, the inter neurons with the ganglion cells, see figure 2.1c. These inter neurons and ganglion cells are cleverly wired in a way that the cells form a logic circuit, thus they already perform basic determinations for shape recognition and movement detection at this early stage of processing. Interested readers may refer to [Becker-Carus, 2004] for more details.



(a) High level view of the Visual System



(b) Anatomy of the Dioptric Apparatus



(c) Schematic of the Retina

Figure 2.1.: Anatomy of the Human Eye, figure (a) shows a high level view of the visual system with indicated direction of signal transport. Figure (b) shows a cross section of the human dioptric apparatus, *Source:* [Knowledgerush.com, 2011]. Figure (c) shows the schematic of the photo receptors and the wired cells of the retina.

The receptors with their first processing layer convert photons into bio- electrical signals, which can be further processed by the brain. There are four types of photo receptors, three types of cones with about 6 million per eye and the rods, which form the majority of receptors with about 120 million per eye.

2.1.2. Response to Colour

Each type of cone has a specific response sensitivity to a small interval of light wave length. The maximum of sensitivities are 419nm for the first type, 530nm and 558nm for the second and third type, respectively. For this reason, the cones are also classified by L(long), M(medium), and S(short) cones, referring to the wavelengths. With the information about the distribution of wavelengths over three sampling values our brain is able to conceive the light colour. However, the cones need a minimum light intensity in order to function, which lies at about the light intensity of bright moonlight or the sky light right after sunset, in numbers this corresponds to $10^{-2} cd\ m^{-2}$. As the light intensity falls underneath that threshold, the fourth kind of receptor, the rods, begin to surrogate the light detection. However completely switching from photopic vision(cones) to scotopic vision(rods) takes up to 30 minutes, this mechanism is called dark adaption. The rods have a much lower response time, which allows humans to perceive higher speeds of objects and higher frequencies of light flare. Also the sensitivity of rods is much higher than that of cones. The maximum wave length sensitivity of rods lies at about 505nm (which corresponds to a colour between green and blue). Because there is only one type of dark light receptor(one wavelength), colours cannot be perceived when the eye is dark adapted. This leads to the overall response of the human eye to wavelengths between 380nm to 760nm. Figure 2.2 shows the segment of wavelength where the eye has its maximum sensitivity. In this figure one curve represents the sensitivity for photonic vision, the other for scotopic vision.

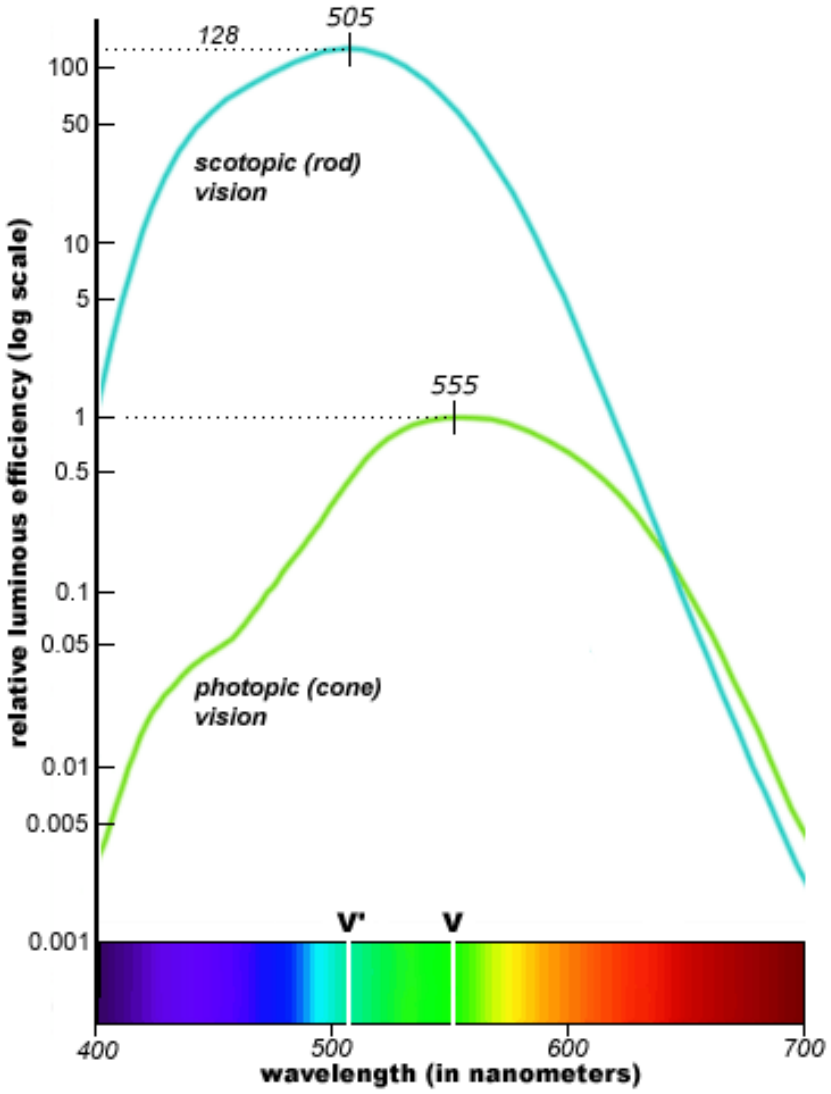


Figure 2.2.: Photopic and Scotopic Response of the Human Eye, *Source:* [Laserpointerforums.com, 2010]

Since there are twice as many cones with maximum sensitivity at green wavelength as the cones for red and blue perception, the combined overall sensitivity lies at about 555nm. One can see the much higher sensitivity to luminous efficiency of the scotopic vision compared to the photopic vision.

2.1.3. Psychological Aspects

Besides the spectral sensitivity to certain wavelengths, the eye, when considered as a complete visual sensor, does much more than just detecting colors. The eye is not only the eye ball itself, but consists of several processing units, for example the inter neurones, ganglion cells and the primary visual cortex, see figure 2.1a. In this context the eye should be seen as a system of physiological detector(the receptors) and a complex psychological processing system(retina and certain parts of the brain such as the visual cortex). This complex system is capable of quickly detecting shapes and edges, recognising spectral composition of light and light alteration over time. One very important concept that is the fundament for many of those features, is the concept of contrast. Although the eye is under certain circumstances even capable of detecting a single photon [Becker-Carus, 2004], the recognition of brightness differences of objects within a scene needs a much higher difference in brightness than just one photon. In other words, the absolute sensitivity follows other rules and has different thresholds than the relative sensitivity. Following the thoughts of [McCartney, 1976], "we distinguish an object from its surroundings because of differences in luminance and chromaticity between various parts of the viewed field. Such differences are usefully expressed as *contrasts*. For our purposes, luminance contrasts are more important than chromaticity factors in determining visibility. Assume that an isolated object is viewed against a uniform, extended background.", then the contrast C is defined by

$$C = \frac{L_0 - L_b}{L_b}. \tag{2.1}$$

Where L_0 and L_b are the levels of luminance of the background(L_b) and the object seen at close distance(L_0), respectively. The *visual threshold of perception*, that is to say the just "perceptible increment of sensation" of the human eye, follows a nonlinear function to changes in stimulus. Weber formulated in 1834 a general law of sensation, he stated: "The increase of stimulus necessary to produce a just perceptible increment of sensation bears a constant ratio to the whole stimulus."[McCartney, 1976]. Later the psycho physicist, Fechner, investigated this law in 1859 and found that, when the field luminance changes from L_1 to L_2, then the perceived change is proportional to $logL_2 - logL_1$, except for very high and very low values of luminance. This law has been investigated by many workers since then, such as Green(1932), Knoll et al.(1946) and Blackwell(1946). The investiga-

tions of Blackwell however are the most outstanding because of its sheer amount of data ascertainment. He presented extensive test data also known as the Tiffany Foundation data, which contain about 450,000 observations that are suitable for statistical analysis collected by 19 observers. The test setup can be summarised as "circular stimuli of various sizes ranging from 0.6 to 360 minutes in angular diameter were presented with various contrasts, positive and negative, on a large uniform background."[Middleton, 1952], measured were the thresholds of luminance-contrast for 50% certainty of detection. For interpreting the data the concept of luminance-contrast threshold will now be introduced, it is common to denote the threshold of luminance-contrast with the Greek letter ϵ. If there are two adjacent objects and they are just distinguishable with the degree of certainty of 50%, and the luminance of one is L and the other is $L \pm \Delta L$, then

$$\epsilon \equiv \left| \frac{(L \pm \Delta L) - L}{L} \right| \equiv \left| \frac{\Delta L}{L} \right| \tag{2.2}$$

where $\frac{\Delta L}{L}$ is the ratio of the least perceptible increment of field illuminance. The data plot that best represents the results of the Tiffany Foundation data is given in figure 2.3. The figure shows the luminance-contrast threshold as a function of field luminance, the curves have been multiplied to bring the probability of detection to about 90%(50% originally).

Very apparent in the graph is the discontinuity in all the curves at about $2 \times 10^{-3} cd/m^2$, which marks the transition from scotopic to photopic vision as referred to in the preceding subsection. Noteworthy here is also the wide range of background luminance of 100 million to 1, giving an also wide variety of contrasts, ranging from 1% to almost 1000. As one can see, the human eye is a remarkable device that is capable of detecting objects precisely over a wide range of contrasts and luminances. Its anatomy has therefore been an inspiring example for the construction of electronic cameras such as CCD and CMOS devices. With the application of visibility improvement in the back of one's mind, the author likes to make a first note at this point, that it is already apparent that possible visbility improvements might follow from figure 2.3. It can be seen from the figure that the human eye has a specific threshold of luminance-contrast, when the contrast of two objects falls below that ϵ, visbility is compromised and the two objects can't be distinguished. With the help of computers, or to be more precise, postprocessing an image by altering the contrast in terms of raising C, one could improve visibility. Especially at the

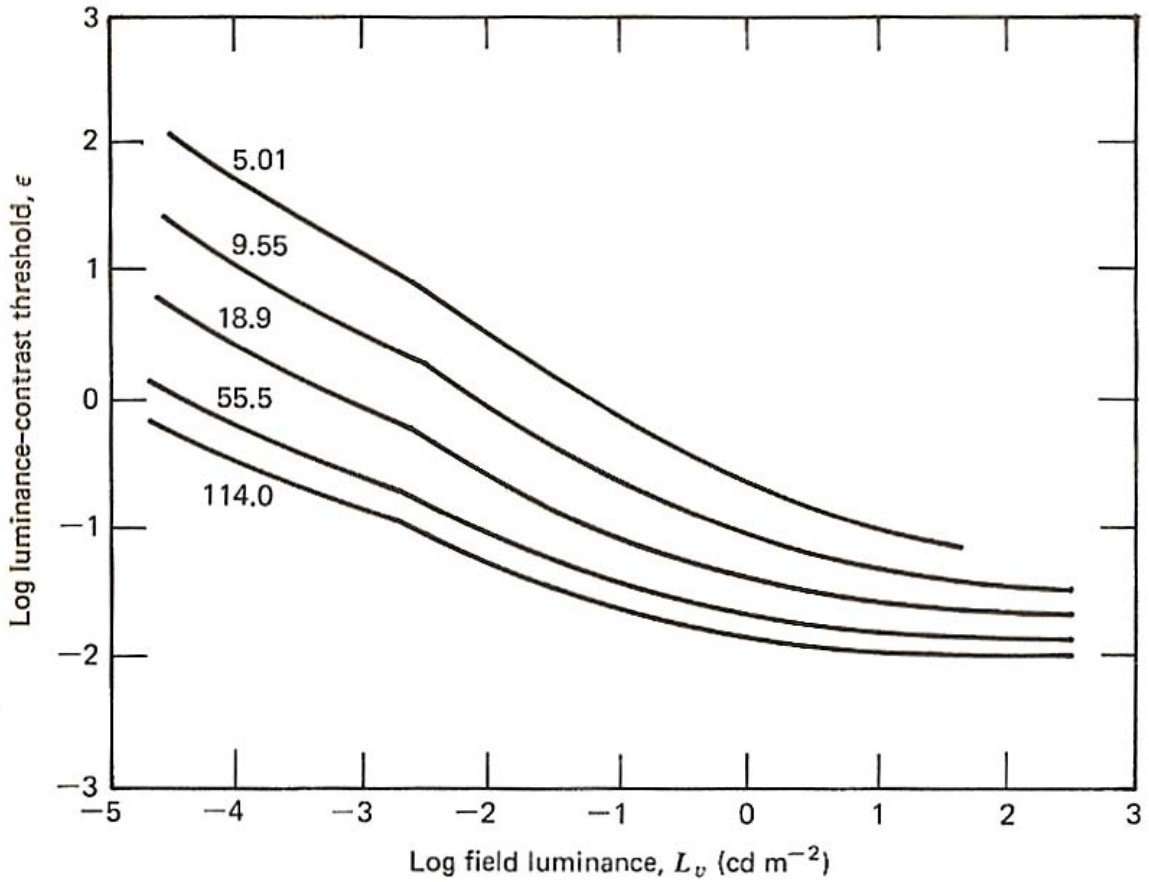


Figure 2.3.: Luminance-contrast threshold as a function of field luminance. The number alongside each curve gives the angular sub tense of the target in arc-minutes. *Source:* [McCartney, 1976]

transition of scotopic to photopic vision, object detection could be improved by using postprocessed images with adequate brightness of objects in relation to other objects within the scene and in respect to the luminance of the surrounding room in which the image is presented(e.g. on a computer monitor). Especially when trying to detect objects in low contrast images, the detection of such objects may be prone to long detection times due to the working principles of the human eye.

2.2. Basics of the Atmosphere

As light travels through a medium, it will most likely get altered due to interactions with the particles of that medium if the travel distance through that medium is sufficiently

long. Reflections, scattering or absorption may happen depending on various factors of the medium. Due to these effects, it is plausible to conclude that light can't travel endlessly through a medium and especially not undistorted. To understand how exactly the atmosphere alters incident light, one must have a closer look at the composition of air and the atmosphere. This section will give a broad overview of the composition of the atmosphere and the origination of particles. The author will conclude the section with an observation of special atmospheric conditions such as fog and clouds.

2.2.1. Optical Concepts in Atmospheric Vision

This subsection shall cover the basic optical concepts that will be used throughout this thesis. The term visibility in this context is synonymous to visual range, which is a measurement being very subjective and varying from person to person. Visual range is defined as the range to where a reference object is just discriminable from the background, or in other words "the distance, under daylight conditions, at which the apparent contrast between a specified type of target and its background (horizon sky) becomes just equal to the threshold contrast of an observer..."[Huschke, 1959]. The visual range therefore depends on the observer(due to the threshold contrast), the size and constitution of the reference object and the background light. The formula for visual range assuming a standard reference object R_v is:

$$R_v = \frac{1}{\beta_{sc}} \ln \frac{C}{\epsilon}. \tag{2.3}$$

Where C is the contrast of the target against the background, ϵ is the threshold contrast of the observer(a basic value can be obtained for example from the Tiffany Foundation data), and β_{sc} is the scattering coefficient. The thesis will discuss the scattering coefficient later in detail in the scattering section. Due to the threshold contrast ϵ, the photographic range is always greater than the visual range, since the photographic range uses the same formula as the R_v but with a minimal ϵ. Humans can only distinct two objects when the contrast is high enough, with photographs however, this contrast can be modified and visibility improved. These subjective factors have been eliminated in the meteorological range R_m. The standardised reference object is a black body radiator that is large enough for the scene and $\epsilon = 0.02$, which is not far from the actual threshold contrast of a human

eye in daylight conditions. Since the target is black, its inherent contrast against the sky is unity, and equation 2.3 becomes

$$R_m = \frac{1}{\beta_{sc}} \ln \frac{C}{0.02} = \frac{3.912}{\beta_{sc}} \tag{2.4}$$

which is, although empiric too, far more convenient. The meteorological range is also called standard visibility or standard visual range. Typical scattering coefficients can be obtained from table 1.1 on page 2.

In aviation meteorology, the runway visual range (RVR) is often used, which defines the distance over which a pilot of an aircraft standing on the centerline of a runway can see the runway surface markings defining the runway or identifying its center line. This is a special case of the visual range R_v.

2.2.2. Composition of the Air

The visibility in atmosphere has been categorised as shown in table 1.1 on page 2. Visibility is among other things dependent on the type of atoms, molecules and particles, in the field of view as well as their number and size distribution. In the simplest of cases, only pure air is present in the observed atmosphere. Although it is convenient for some research issues to introduce the concept of pure air, this is only a theoretical construct that marks the upper boundary of visibility in the atmosphere. Pure air only contains molecules with molecule sizes of $10^{-4}\mu$m in radius and a concentration of about 10^{19} per cm^3. However, these are not the only particles present in the atmosphere, all types of particles responsible for atmospheric scattering are shown in table 2.1.

Table 2.1.: Particles Responsible for Atmospheric Scattering

Type	Radius(μm)	Concentration(cm^{-3})
Air molecule	10^{-4}	10^{19}
Aitken nucleus	$10^{-3} - 10^{-2}$	$10^{4} - 10^{2}$
Haze particle	$10^{-2} - 1$	$10^{3} - 10$
Fog droplet	1 - 10	100 - 10
Cloud Droplet	1 - 10	300 - 10
Raindrop	$10^{2} - 10^{4}$	$10^{-2} - 10^{-5}$

Source: [McCartney, 1976]

Also, the concentration of gases in the atmosphere is not constant. The atmosphere, that is to say the overall envelope of the earth is several hundred kilometers thick, gases and therefore optical characteristics alter greatly with altitude. Hence this work will restrict the altitude of interest to the layer directly above the ground and just a few kilometers above in order to confine the work to just what is needed for the desired application of image dehazing near the ground. The gas concentrations of the permanent gases can be considered constant within one layer of altitude according to [McCartney, 1976] and [Middleton, 1952]. An excerpt of the most important permanent gas constituents can be found in table 2.2. It is noticeable that the two gases nitrogen and oxygen make up for about 99% of all the gases in the atmosphere near the ground. Many gases only occur as traces.

Table 2.2.: Atmospheric Gases Present in Permanent Amounts

Constituent	Volume ratio	Parts per million
Nitrogen, N_2	78.08 %	-
Oxygen, O_2	20.95 %	-
Argon, Ar	0.93 %	-
Carbon Dioxide, CO_2	0.037 %	-
Neon, Ne	-	18
Helium, He	-	5
Methane, CH_4	-	1.7
Hydrogen, H_2	-	0.6
Nitrous oxide, N_2O	-	0.3
Xenon, Xe	-	0.09

Source: [Essentials of Physical Geography, 2006]

There are variable amounts of gases as well, as shown in table 2.3. Regardless of their small absolute volumes, ozone, vaporised water and water play an important role in atmospheric optics, because of their strong absorption of ultraviolet and infrared, respectively. Additional importance is attached to water, because of its influence on the growth behaviour of particles in the atmosphere, which will be dealt with later in this section.

Table 2.3.: Atmospheric Gases Present in Variable Amounts

Constituent	Volume ratio	Parts per million
Ozone, O_3	-	0-0.07 (ground)
	-	1-3 (20-30km)
Water vapour, H_2O	0-2	-
Nitric acid vapour, HNO_3	-	$(0-10)\times10^{-3}$
Ammonia, NH_3	-	Trace
Hydrogen sulfide, H_2S	-	$(2-20)\times10^{-3}$
Sulfur dioxide, SO_2	-	$(0-20)\times10^{-3}$
Nitrogen dioxide, NO_2	-	Trace
Nitric oxide, NO	-	Trace

Source: [McCartney, 1976]

In regions of pollution, especially in industrial regions, other particles may be present depending on the type of local industry, this must be treated separately. Apart from these obvious regions where particles are present in the atmospheric air, other regions are filled with dust as well, in fact the air is never free of particles. They have a broad variety of origins, sizes and chemical compositions. Special interest here lies on those particles who are good scatterers. These can be cosmic dust, volcanic ash, foliage exudations, combustion products, bits of sea salt and many more. Dust grains and hygroscopic particles whose size depends on the relative humidity, as we will see later in more detail, are of special interest to this subject. These are large particles and the larger a particle is, the better it scatters light. The particles of interest range from about 0.01 to 10μm. These particles form a dispersed system suspended in gas, called *aerosol*. The term *haze aerosol* states the particle nature of haze. Citing McCartney, "From the optical standpoint, haze is a condition wherein the scattering property of the atmosphere is greater than that attributable to the gas molecules but is less than that of fog. Haze scattering imparts a distinctive gray hue to the sky, which would otherwise appear a deep blue, and is usually the determining factor for visibility."[McCartney, 1976].

Large amounts of dust get continually suspended in gas by several natural mechanics. One of the most important processes, not just for aviation, are volcanic activities. The ash that is catapulted into high altitudes where particles can remain several years in the atmosphere and get transported several thousand kilometers. Especially through the stratosphere, where the particles can spread before they eventually settle by wash out processes and gravitation, respectively. This was investigated by Cadle(1966) and others.

Also considerable are the interplanetary debris swept up by the earth travelling through space. Petterson(1960) estimated that about 10^7 tons of this material could be collected this way in a single year.

Surface winds are very effective in loosening, lifting and transporting large amounts of dust, especially over large dry surfaces. Winds already containing grains of dust may loosen even more material by erosion. The direct effects of surface dust can be seen locally, when the surface winds may form clouds of dust as in sand storms. One prominent member of these particles may also be loess.

Men too, is responsible for several causes of dust suspension, as he builds industries and construction sites. Both are sources for dust and responsible for its lifting for example by chimneys. Depending on the size of the particles they may affect just the local area around the sources or if the particles are sufficiently small(smaller than 1μm) they move with the winds for several days, where they remain suspended only until the first washout.

Very important scatterers are the hygroscopic particles due to their ability to grow in size by collecting water, as they function as condensation nuclei. Like the dust particles, condensation nuclei have several possible origins, however they differ from the dust particle origins. "In contrast to the aridy implied by a dust cloud, the green world of plants and trees creates its own organic type of aerosol, described by Went (1955) as summer *heat haze*."[McCartney, 1976], these organic aerosols with an average radius of about 0.15μm are very small and naturally found where large quantities of plants occur, such as the tropical rain forests or timbered slopes of mountain forests. Besides organic origins there are several other sources, as described next.

All combustion processes create many small particles. They may chemically react with other materials in the atmosphere, which can form additional kinds of gases or new small particles. In industrial and metropolitan regions where combustion engines accumulate the result is a polluted atmosphere that bears a dark haze which can even

be seen from an aircraft several kilometers away.

But not just industries are responsible for smoke particles, ordinary grass fires or forest fires produce vast amounts of small particles with about 0.1μm in radius that are hygroscopic.

Some types are produced by photochemical reactions and nucleations between combustion gases and atmospheric trace gases which may originate also from volcanic eruptions, as described by Manson et al. (1961).

Finally, sea salt particles, being highly hygroscopic, are very important components of maritime aerosols. Winds picking up small water droplets from open waters may evaporate the water and suspend the salt elements, transporting them over great distances. Each bit of salt functions as condensation nuclei, varying in shape and size depending on the relative humidity. As an example how important these particles are to visibility, Wright (1940), who studied haze aerosols of mixed composition at several sites in England for many years, found correlations between relative humidity and meteorological range, the latter decreasing sharply at relative humidities greater than 65%. He states that salt particles, rather than combustion nuclei play the predominant role in haze and fog in England.

With respect to airports and aviation in general, the following particles can be expected as reported by the IPCC(Intergovernmental Panel on Climate Change): "Aircraft engines also directly emit soot and metal particles. Liquid aerosol precursors include water vapor, oxidized sulfur in various forms, chemi-ions (charged molecules), nitrogen oxides, and unburned hydrocarbons. Large numbers (about 10^7/kg fuel) of small (radius 1 to 10 nm) volatile particles are formed in the exhaust plumes of cruising aircraft, as shown by in situ observations and model calculations."[IPCC, 2000], although absolute numbers of exhaust particles are not validated against the total number of particles in a unit volume of atmosphere here, it should be apparent from this statement, that in regions of airports these particles are present in larger numbers compared to the country side and other regions.

2.2.3. Particle Size Distribution

As stated in the preceding sub section, condensation nuclei, especially salt particles may grow in size depending on environmental factors such as the relative humidity. Dessens

investigated the size and shape of haze particles, he was the first who was able to capture single haze particles and photograph them without compromising their shape. Even more astonishingly, he developed a method to study their response to changes in humidity. Haze particles are very small and leave no trace on ordinary surfaces, hence they were considered unobservable for a long time. Whereas other workers tried to capture small particles and droplets on small slides covered with oil(Bricard) or Vaseline(Houghton and Radford) where the particles may change their original state to a spheric shape, Dessens had the ingenious idea to use spider webs, not just from common spiders, but webs from exceptionally small spiders that produce filaments of the order of just $10^{-6}cm$, which is invisible under a normal optical microscope. Only in presence of particles they become visible. The samples caught this way could be kept indefinitely in a closed box. Dessens photographs through the microscope of one sample can be seen in figure 2.4, the filaments were rendered visible later.

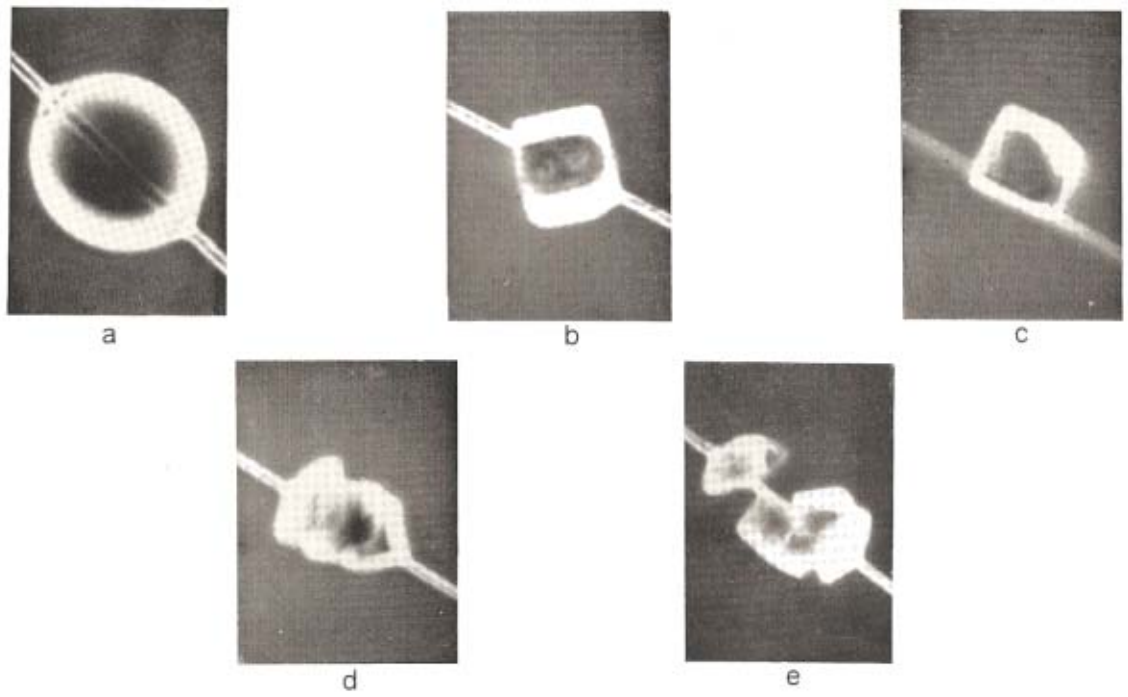


Figure 2.4.: Forms assumed by a large haze particle, *Source:* [Dessens, 1946]

The droplet shown in figure 2.4 was captured and used for dozens of operations on this particle. Figure 2.4 (a), remained stable for one hour, then it changed abruptly to a fine crystal (b). Dessens was able to change it from a droplet to a crystal (b), (c), or a group

of crystals (d), (e) and back again at will, by changing the relative humidity h. However, the behaviour is much more complex than this would indicate. Dessens insights may be presented better in the following way, as it was done by [Middleton, 1952]:

"(1) He could always pass from crystal to droplet by simply breathing on the crystal; but

(2) The passage from droplet to crystal is possible in general only if $h > 65$ per cent.

(3) If, with $h \equiv 50$ per cent, he breathed on the crystal until it grew to more than a certain radius 9μm for this drop and then stopped, it evaporated to the critical radius in less than a second, remained unchanged for ten seconds or so, and then abruptly changed into a crystal in a second or less.

(4) If on the other hand the droplet as he breathed on it did not reach the critical radius, the change to a crystal occurred immediately the source of moisture was removed.

(5) The time during which the drop remained "metastable" increased with relative humidity, and could be hours or days at h=65 per cent to 70 per cent."

In terms of visibility, the important result from Dessens spider web experiment is, that haze particles may change in size and shape. It should be clear from basic optical physics that the constitution of a medium with its refractive index, shape of surface and thickness affects the incident light greatly. Also, haze particles even considered very locally are not homogeneous in terms of size, "Hazes are polydisperse aerosols in which particle sizes vary across two and three orders of magnitude"[McCartney, 1976]. Due to streams in the air, collisions and changes in relative humidity, they will never have the same size and shape in practice. As stated earlier however, the size is very important to its optical effects, hence different techniques have been employed to count and measure particles. Typically sizes are divided into size classes, which usually are 0.1 to 0.5μm wide or greater if fog and clouds are examined. Such data can be represented as histograms which give the population n_i of each size class r_i, an example is given in figure 2.5. A unit volume of aerosol is implied. The overall concentration N, which is the total number of particles per unit volume, is equal to the sum of the class populations, that is

$$N = \sum_{i=0}^{k} n_i(r_i). \tag{2.5}$$

Beside the concentration N, a function that characterises the size distribution is desirable, one that is adaptable to different scenarios such as continental or maritime haze and cloud and fog, respectively. Several size distribution formulas have been devel-

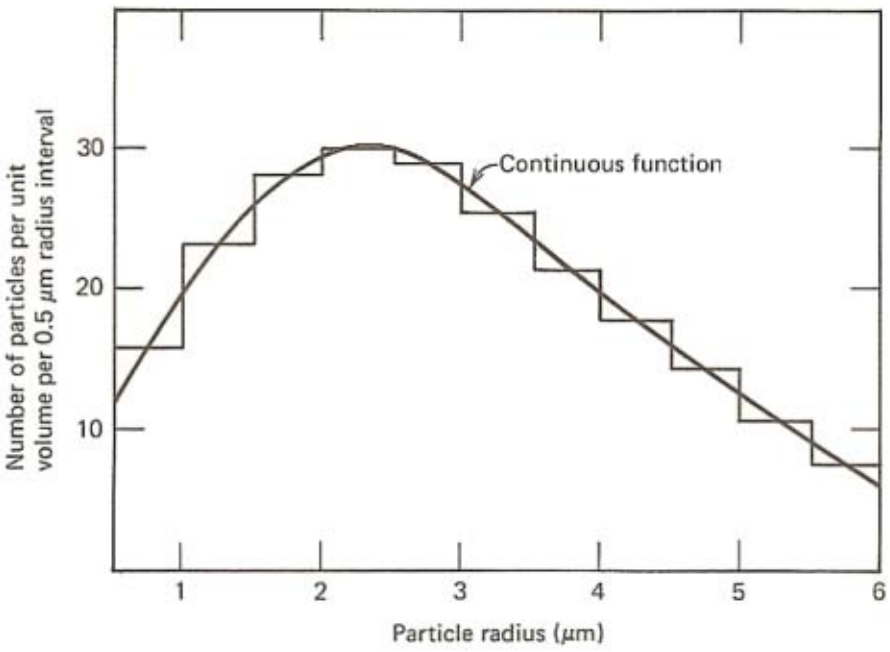


Figure 2.5.: Histogram of particle size data. *Source:* [McCartney, 1976]

oped by different researchers, such as Deirmendjian(1963), Junge(1960) and Clark and Whitby(1967). One formula has been proven as especially helpful for haze investigations, the power-law size distribution function developed by Junge(1960). It is

$$n(r) = \frac{dN}{d\log r} = cr^{-\upsilon} \tag{2.6}$$

where N is the number of particles per unit volume, from the smallest size up to size r, c is a constant accounting for the concentration, and finally the exponent υ determines the slope of the distribution curve. Since the derivative is with respect to $\log r$, dN is the number of particles per increment $\log r$. A plot from actual measurements using the power-law size distribution equation is shown in figure 2.6.

The power-law can also be put into a non logarithmic form, with
$d\log r = 0.434\ d\ln r$, and
$d\ln r = \frac{dr}{r}$.
Equation 2.6 can then be written as:

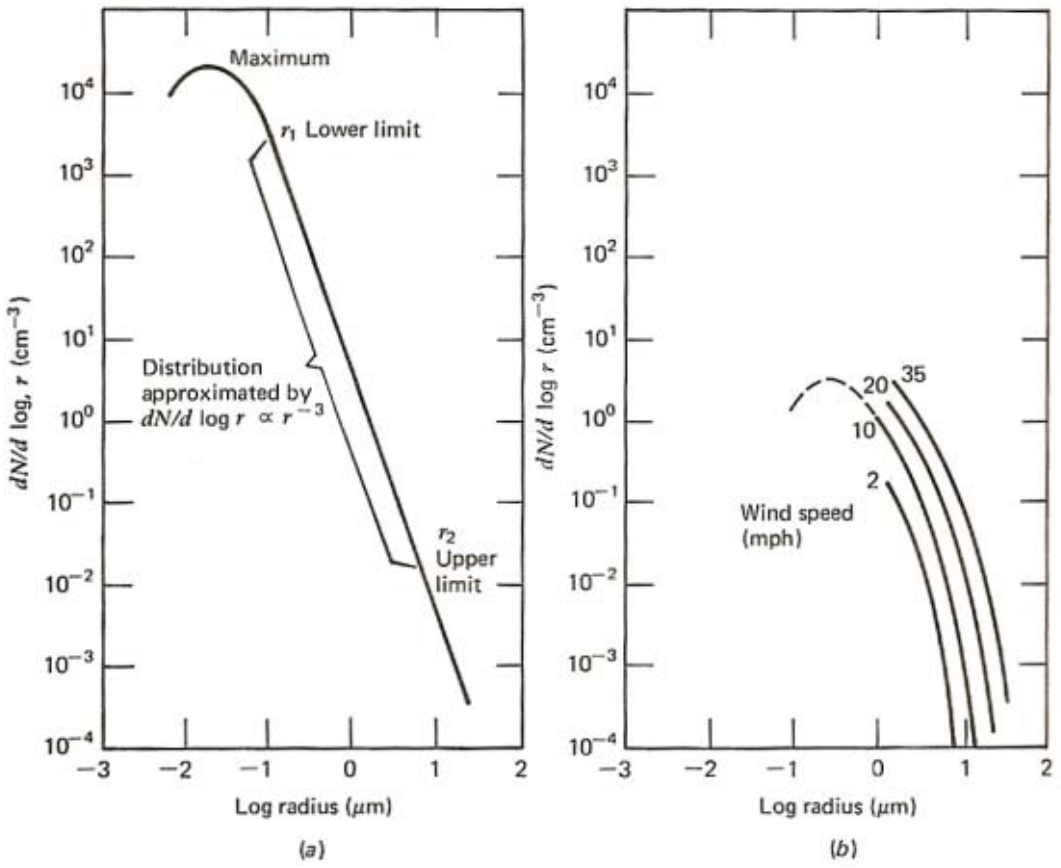


Figure 2.6.: Power-law size distribution of particles in continental and maritime aerosols according to the power law, equation 2.6. (a) continental type, (b) maritime type. *Source:* [Junge, 1960]

$$\widetilde{n}(r) = \frac{dN}{dr} = 0.434cr^{-(\upsilon+1)} \tag{2.7}$$

It can be seen that the amount of larger particles, relative to the amount of smaller particles increases as the value of the exponent decreases. Typical hazes are best described by $3 > \upsilon > 4$. While fogs are characterised by $\upsilon \approx 2$. The particle concentration can be obtained from integrating equation 2.7:

$$N = 0.434c \int_{r_1}^{r_2} r^{-(\upsilon+1)} dr. \tag{2.8}$$

Note that, when $n(r), \upsilon$ and N are found experimentally from counting, c can be deter-

mined from equations 2.6 and 2.8. Comparing the two maritime and continental curves in figure 2.6, the greater concentration and smaller size of particles of the latter type are noteworthy. The difference between maritime and continental aerosols mostly comes from the predominance of sea salt nuclei in maritime aerosols, they tend to have fewer but larger particles due to their hygroscopic characteristics and are influenced by wind speed as shown in figure 2.6(b). Of course there are no sharp distinctions between the two, because continental and oceanic air masses are not kept apart by coastlines. It is also noteworthy that actual particle size distributions on an individual basis can differ greatly from the strict power-law form. On the average however, the power law seems to be a good representation of aerosols in general. It would be impractical to have one function for each type of aerosol, however specific sets of constants can be applied to the general forumula to simplify the calculations for specific aerosols. The formula has been evaluated by several workers by measurements in the field, among others Whitby(1971) investigated the aerosols of Los Angeles and found that in the size range of 0.01 to 2 μm, there was a dependency of $n(r)$ to r^{-4} in most cases. Clark and Whitby(1967) found a simplification from experiments, that states:

$$n(r) = 0.05\phi r^{-4} \tag{2.9}$$

with r in micrometers. Which is equivalent to the non logarithmic power law function(2.7) with $\upsilon = 3$ and ϕ expresses the concentration of particles in cubic micrometers per cubic centimeter of aerosol. The product 0.05ϕ corresponds to c in equation 2.7. This relationship was valid for particles from $r = 0.05\mu$m to 3μm. From averaged data, a representative value of ϕ was found, such that

$$n(r) = 2.49r^{-4} \tag{2.10}$$

This is an example of how convenient the power-law equation is and how it can be adapted to actual measured data. However this equation meets only the restrictions for a horizontal path of sight since particle concentration and size distribution varies with altitude. Formulas that take into account the altitude are far more complex and due to the constraints made for the subject of dehazing are not necessary to investigate in this thesis.

2.2.4. Relative Humidity

As described in the preceding subsection, relative humidity has a big influence on particle sizes of hygroscopic nuclei. For better understanding, this subsection will describe the concept of relative humidity in more detail since it will be used in later sections as well. The humidity of the air denotes the proportion of vaporised water in the air compound. Hence liquid or solid forms water are not measured for air humidity. Humidity is an important parameter for meteorologic and technical processes, for this purpose however it is an important factor for visibility through haze, fog and clouds, respectively. At a certain temperature and pressure, only so much vaporised water can be contained by a unit volume of air. The most common measure for humidity is the relative humidity that is given in per cent. It denotes the proportion of the current amount of water vapour to the maximum possible amount of water vapour at the same air pressure and temperature. The formula for relative humidity therefore is:

$$relative\ humidity[\%] = \frac{current\ absolute\ humidity[g/m^3]}{maximal\ absolute\ humidity[g/m^3]} \times 100. \tag{2.11}$$

Although there is no compulsory symbol, relative humidity is often written as the Greek letter ϕ, thus the equation can be simply written as:

$$\phi = \frac{s}{S} \times 100\%. \tag{2.12}$$

With s the specific humidity and S the saturation humidity. As stated earlier, hygroscopic nuclei grow in size with relative humidity, since bigger particles are able to scatter light better than small particles, this results in a dependency of visual range to relative humidity. Figure 2.7 shows this correlation. The curve is an average of many observations at the Los Angeles airport, a region that is very rich of nuclei.

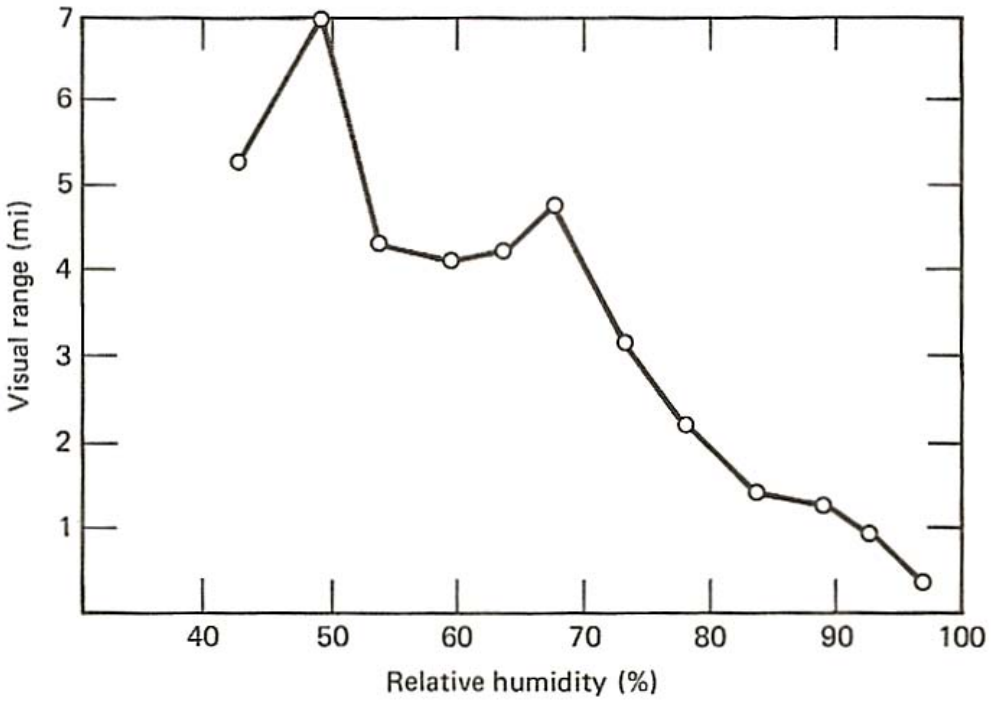


Figure 2.7.: Visual range as a function of relative humidity at Los Angeles airport. *Source:* [Neiburger and Wurtele, 1949] and [George, 1951]

2.2.5. Fog, Clouds and Smoke

Along with haze and dust, the latter usually implying that the air is dry, there are other hygroscopic and non hygroscopic objects which will be the topic of this subsection. The distinction between the humid phenomena can be made through the droplet size and concentration as it is shown in table 2.1 on page 15. As a rule of thumb, the higher the concentration and the bigger the particles are, the more they will scatter light, hence the visual range will drop. Although water absorbs certain wavelengths, it does not absorb all light, however some particles, especially black smoke particles are able to absorb visible light independent from wavelength. This will of course result in a strongly decreased visual range. These particles would need further classification. It is also noteworthy that clouds are not the same as fog, the visual effects are of different magnitude as are the constituents in terms of size, size distribution and number. To visualise the droplet size, one should have a look at figure 2.8, it shows the differences in particle sizes of cloud, fog and haze drawn to the same scale.

These photographs however only show certain types of clouds and fogs, the fog shown in

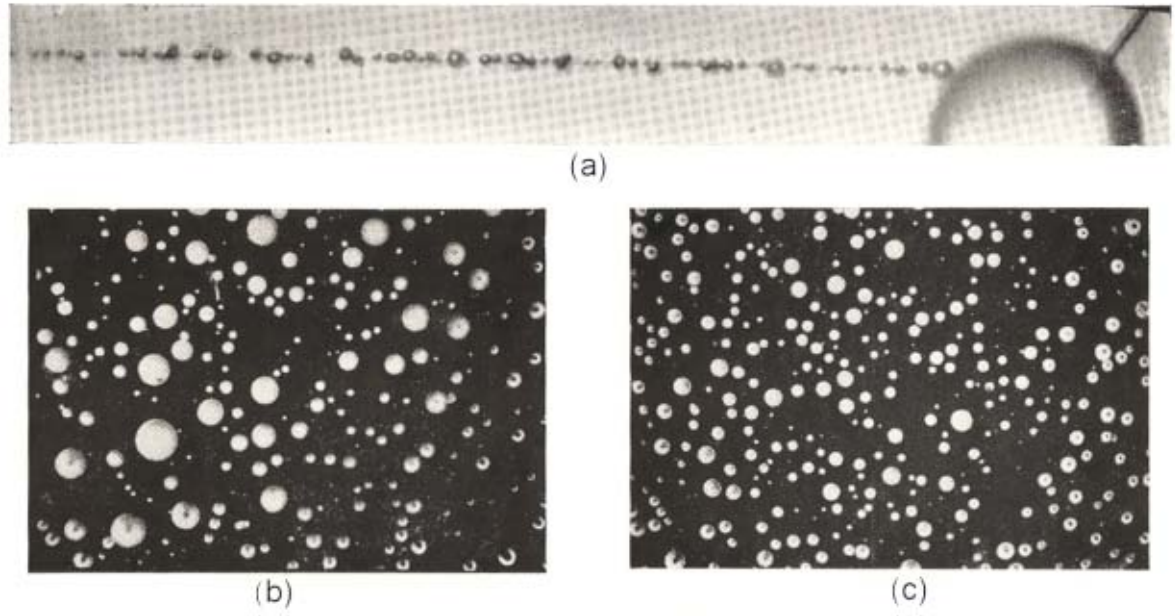


Figure 2.8.: Photographs of Cloud, Fog and Haze Droplets. (a) shows the photomicrograph of haze particles, *Source:* [Dessens, 1946]. (b) shows a photomicrograph of typical fog particles, *Source:* [Houghton, 1939]. (c) shows a photomicrograph of typical cloud particles, *Source:* [Houghton, 1939]

figure 2.8 is a coastal fog with larger droplets than the cloud and also a larger range of radii. Clouds also are not always of the same type, since they may have different origins and altitudes. Common mean radii of clouds depend on their type: Nimbostratus(10μm), Stratocumulus(8μm), Cumulus(5μm) and Stratus(4μm). However all these particles and cloud types, even rain could be described by the power-law size distribution function, described in the earlier subsection. This shall just show the large variety of particles in the air and their constitution and origin, respectively. It is therefore a complex problem to find a dehazing algorithm that can handle all kinds of haze, clouds, fog and smoke, respectively. The formation of clouds and fog however are not topic of this thesis. At this point, the reader should have enough basic knowledge to understand the upcoming core principles such as light scattering and the "Theorie der horizontalen Sichtweite" after Koschmieder. The author wishes to address only one more topic before the thesis will carry on with the core principles.

As it was addressed by Middleton in 1952 already, there is a myth of a diffuse boundary of fog and smoke. A common belief is, that fog inherits the ability to distort shapes/outlines. Such that, for example the outline of a person seen through a wall of fog, is horizontally or otherwise distorted and that the actual outline can not be seen. There is no scientific

data that shows this phenomenon. All photographs that show objects through thick fog either show clear outlines of that object or no object due to light extinction. It is more a stylistic device of painters than a real world effect. This can also be shown by light beams, for example that of a flashlight seen in fog or smoke, although the light appears weaker along the path as it travels through the medium, the boundaries of the cone or beam are not diffuse but indeed sharp. This is a fortunate characteristic and therefore does not need further regard from the dehazing algorithm.

2.3. Light Scattering

Light scattering is the most important phenomenon to this subject. It describes the interaction between a (haze) particle and a photon. This section will deal with light scattering in detail, initially from a general perspective but then in subsequent sections with respect to Rayleigh scattering and Mie scattering. But first, a motivation for scattering research is given.

2.3.1. Motivation

In a common scenario where air molecules are in a concentration of $N_L = 2.687 \times 10^{19} cm^{-3}$ (Loschmidt's number), the average spacing available to each molecule is $\frac{1}{N_L} cm^3$. Hence the average spacing between air molecule centers is

$$S = \sqrt[3]{\frac{1}{N_L}} \approx 3.3 \times 10^{-7} cm \approx 33\text{Å} \tag{2.13}$$

which is about 9 times the diameter. The mean free path $\bar{l}$, usually used to express the average distance a molecule travels between collisions is given by

$$\bar{l} = \frac{1}{\sqrt{2}\pi d^2 N_L} \tag{2.14}$$

this adds up to about 600Å when using the effective collision area of $A = \pi d^2$ with the

diameter for air molecules. However to show that a photon will in most cases collide with an air molecule or haze particle in distances common for visual ranges, one can use the effective collision area of a photon with an air molecule instead. A photon, not having a radius, causes the effective collision area to be just $A = \pi r^2$. With r the radius of an air molecule, one will get for the mean free path:

$$\bar{l} = \frac{1}{\sqrt{2}\pi r^2 N_L} \approx 2.4 \times 10^{-5} cm \approx 2400\text{Å} \tag{2.15}$$

This shows that the distance traveled by a photon through air is, although considerably longer than that of an air molecule, extremely small compared to visual ranges. This should be motivation enough to take a closer look at the scattering process itself.

2.3.2. General Considerations of Light Scattering

As a rule of thumb, it can be said that, the larger a particle is, the better it scatters light. This phenomenon is illustrated by figure 2.9, the figures show the scattering pattern of a light beam scattered by 3 different sizes of particles (a) shows the symmetric scattering pattern of a particle smaller than one-tenth of the wavelength of the incident light beam. (b) and (c) show the scattering patterns of larger particles, note that the complexity of the pattern grows with the particle size. The similarity of all scattering patterns of particles larger than one-tenth of the wavelength of the light is that most photons are scattered in forward direction, this effect is increased also with particle size.

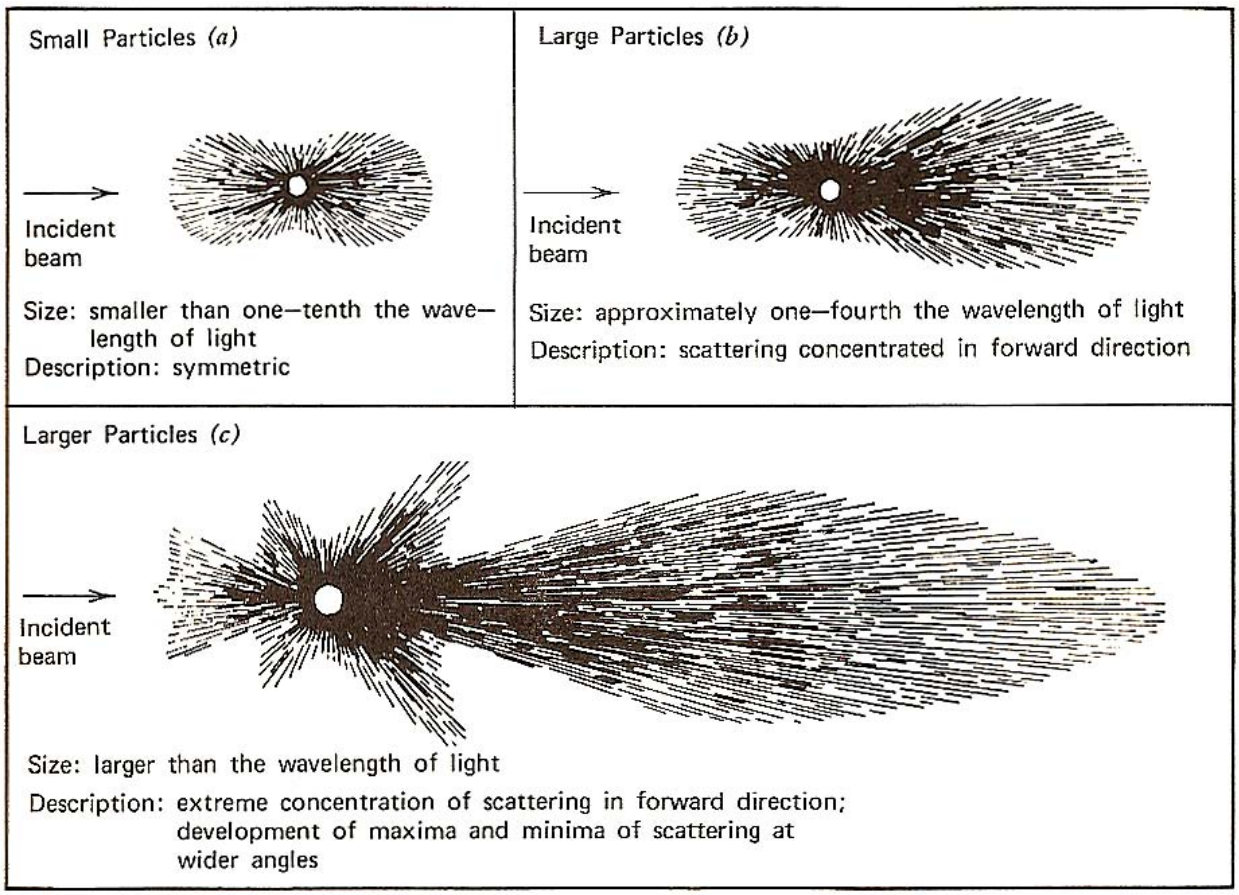


Figure 2.9.: Angular patterns of scattered intensity from particles of three sizes. (a) Small particles, (b) large particles, (c) larger particles, *Source:* [Brumberger et al., 1968]

Because of the large magnitude of visual ranges compared to that of the mean free path, a photon can get scattered by more than one particle along the way between the object and the eye of the observer, this is called multi scattering or rescattering, respectively. However, throughout this section single scattering is considered first in order to describe the basic mechanisms of scattering. There are two main theories that explain light scattering, they are named after their originators, the Mie scattering and the Rayleigh scattering. They both describe the phenomenon of light scattering, however assuming different sizes of particles. Whereas the Mie theory can be used to describe scattering by larger particles such as haze, Rayleigh scattering can be used to explain, among other things, the blueness of the sky due to air molecule scattering. Scattering is an important effect of the atmosphere and all particles contained within, it delimits the visual range. It was attempted to quantify this by introducing the scattering coefficient

β. The scattering coefficient is a part of the extinction coefficient σ:

$$\sigma = \beta + k \tag{2.16}$$

with k being the absorption coefficient. For now however, consider an absorption free atmosphere, meaning the total flux of the incident beam is equal to the total flux of the scattered light in all directions. The scattering function $\beta(\phi)$, which dependents on the observed angle in relation to the incident beam, denotes how much of the incident beam will get scattered in the direction with the angle ϕ, now following the thoughts of [Middleton, 1952], the scattering function is contained in the equation:

$$I(\phi) = Eds \cdot \beta(\phi). \tag{2.17}$$

With $I(\phi)$, the light intensity produced by the incident beam and E, the illuminance at ds. In the absence of absorption it is:

$$\int_0^{4\pi} I(\phi) d\omega = E\ ds \tag{2.18}$$

or,

$$2\pi \int_0^{\pi} E\ ds\beta \cdot (\phi) \sin\phi\ d\phi = E\ ds \tag{2.19}$$

and therefore

$$2\pi \int_0^{\pi} \beta(\phi) \sin\phi\ d\phi = 1. \tag{2.20}$$

Likewise the volume scattering function $\beta'(\phi)$ is:

$$2\pi \int_0^{\pi} \beta'(\phi) \sin\phi\ d\phi = \beta. \tag{2.21}$$

This relation is useful when the actual scattering material is unknown. The scattering coefficient will be used in the following subsections for Mie and Rayleigh scattering. Actual scattering coefficients for various visual ranges can be found in table 1.1 on page 2. It will now be shown how the scattering actually works, first for very small particles(Rayleigh scattering) and then for larger particles(Mie scattering).

2.3.3. Rayleigh Scattering

Rayleigh Scattering is named after the British physicist Lord Rayleigh and describes the elastic scattering of light. The light is considered as an electromagnetic wave scattered by particles much smaller than the wavelength of the radiation. The theory holds for transparent media of all aggregate states. The ratio α, which describes the relation between the wavelength of the incident light and the particle radius, can be used to delimit the transition between Rayleigh scattering and Mie scattering. "The joint importance of wavelength and particle size may be appreciated from the fact that these two parameters determine the distribution of phase over the particle."[McCartney, 1976]

$$\alpha = \frac{2\pi r}{\lambda} \tag{2.22}$$

If $\alpha \ll 1$, then the model of Rayleigh scattering can be applied, for values of $\alpha = 1$ and greater, the Mie theory must be used. However, for small α the Mie Theory reduces to the Rayleigh approximation. When analysing the reaction of a single particle to an incident beam of light, one must consider the particle as harmonic oscillator as part of the oscillator model for elemental scatterers. The key idea of this model is that the incident radiation is received by the oscillator and will emit radiation(a secondary wave) in different directions as though the primary wave were not present. An idealised molecule is similar to the model used in elementary kinetic theory. This means that the molecular mass resides almost entirely in the center, which in this case carries all the net positive charge. The negative charge resides on the outer shell of the ball that is the oscillator. An illustration can be obtained from figure 2.10, the charges are connected via an elastic binding force, characterised as springs making it a dipole oscillator. The model is assumed to be nonionised, nonpolar, isotropic, linear and lightly damped. These assumptions are not far from an actual molecule as for example oxygen or nitrogen, respectively.

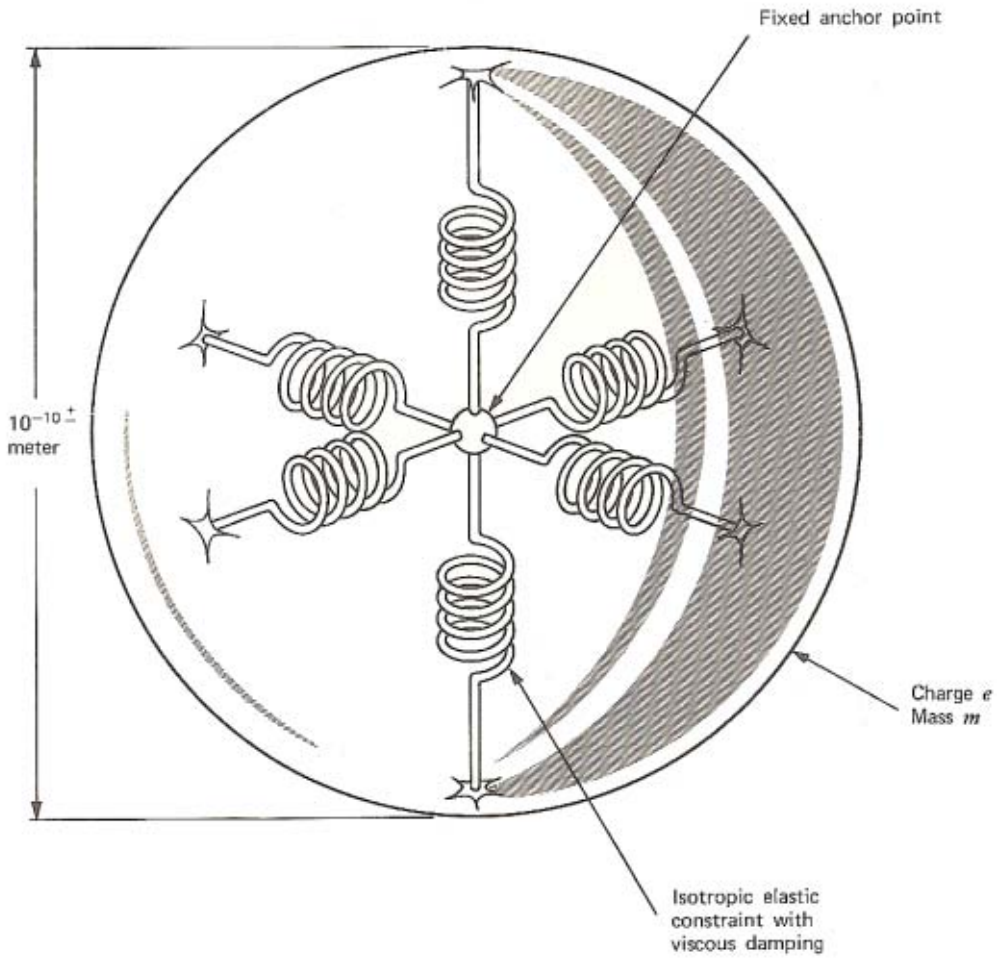


Figure 2.10.: Model of an elemental scatterer, adapted from [Strong, 1958], Copyright © 1958.

Due to the spring constant k, which accounts for the restoring force per unit displacement and the mass m, the oscillator vibrates as a reaction to the incident radiation stimulation with a resonant angular frequency ω_0 of

$$\omega_0 = \sqrt{\frac{k}{m}}. \tag{2.23}$$

Note that the central mass, being much greater than the electron mass, is fixed while only the outer part vibrates. Further investigations of the oscillator model are not necessary for the subject of this thesis. The important conclusion here is that the scattered intensity $I(\phi)$ of light scattered by a single particle from an unpolarised beam of light

with wavelength λ and intensity I_0 is given by the equation:

$$I(\phi) = I_0 \frac{1 + \cos^2 \phi}{2R^2} \left(\frac{2\pi}{\lambda}\right)^4 \left(\frac{n^2 - 1}{n^2 + 2}\right)^2 \cdot r^6 \tag{2.24}$$

with n being the refractive index of the particle and R the distance from the particle in the angle of ϕ. This means that the scattered intensity varies as the inverse fourth power of the wavelength and the sixth power of the particle size, as r denotes the radius of the scatterer. Also very important is the relation of the intensity to the angle ϕ, as this is proportional to $1 + cos^2(\phi)$. The direct effect of this is that the larger the wavelength is, the more of the scattered light will be scattered in forward or backward direction, respectively. However, the pure amount of scattered light as part of the incident beam is strongly dependent on the wavelength, this phenomena is shown in figure 2.11.

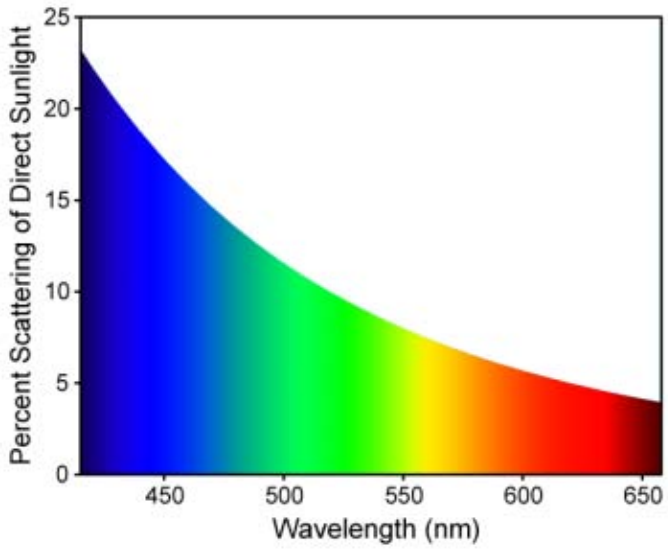


Figure 2.11.: Figure shows the percent of the incident light that is scattered, *Source:* [wikipedia.org, 2011a].

Therefore smaller wavelength are scattered better. Apparent from equation 2.24 is also that blue light(small wavelength) will tendentially be scattered more into 90°(270°, respectively) direction relative to the incident light in the plane of observation and red light will be scattered into forward(0°) and backward direction(180°) in the plane of observation. The more the light is scattered, the more it gets polarised too. Important for visual ranges, where multi scattering inevitably occurs along the path, is the formula

that takes multiple molecules into consideration, the scattered intensity then becomes:

$$I(\phi) = \frac{9\pi^2\ \epsilon_0 c\ \sin^2\phi}{2N^2\lambda^4}\left(\frac{n^2-1}{n^2+2}\right)^2 {E_0}^2 \tag{2.25}$$

where N is the number of dipole oscillators per unit volume in standard air with standard temperature and pressure, ϵ_0 denotes the electric constant, c the speed of light in free space and E_0 the electric field of the incident wave. In practice the refractive index increases somewhat with frequency, so that the wavelength dependence of scattering is slightly greater than λ^{-4}. According to Middleton(1952), the dependence is about $\lambda^{-4.08}$ in the visual region of light. As introduced in the general considerations to light scattering, a convenient way to describe scattering is the scattering coefficient β. It is both common to use the angular and total scattering coefficient as it will be described next. The coefficients always account for the volume, meaning the additive scattered fluxes from many molecules. The angular scattering coefficients account for the polarisation of the incident light, hence for polarised light $\beta(\phi)$ is

$$\beta(\phi) = \frac{16\pi^4\ r^6\ N}{\lambda^4}\left(\frac{n^2-1}{n^2+2}\right)^2 \sin^2\ \phi \tag{2.26}$$

but the formula for unpolarised light, which can be assumed for sunlight, becomes:

$$\beta(\phi) = \frac{8\pi^4\ r^6\ N}{\lambda^4}\left(\frac{n^2-1}{n^2+2}\right)^2 (1+\cos^2\ \phi) \tag{2.27}$$

The difference comes from the polarisability of the particle itself which is a purely physical effect and will not be considered further here. The scattering pattern implied by the equations can be found in figure 2.12.

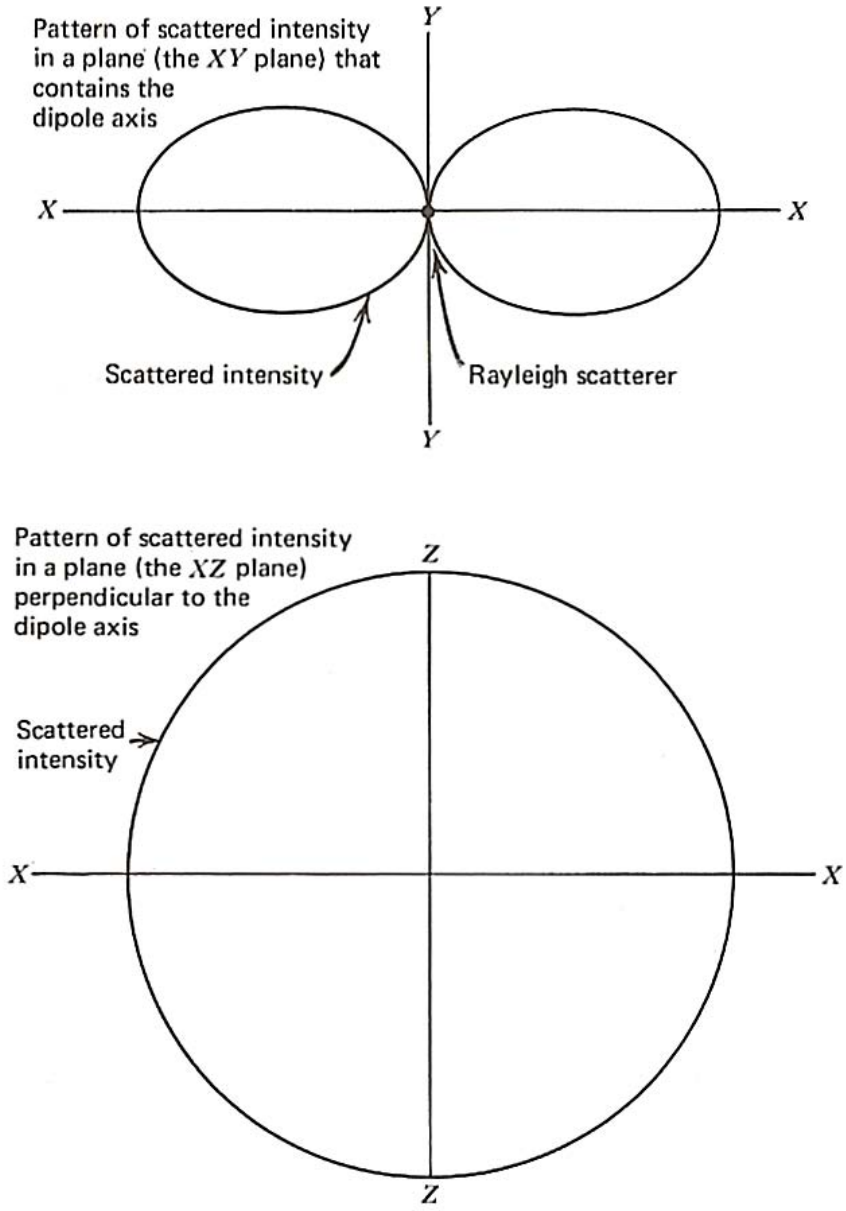


Figure 2.12.: Angular scattering intensity of a Rayleigh scatterer, top figure shows scattering intensity in XY plane, bottom figure shows the scattering in XZ plane, *Source:* [McCartney, 1976]

These equations are very similar to those of the scattered intensity, and as stated by McCartney: "When the incident flux has unit irradiance, the numerical value of $\beta(\phi)$ represents the scattered intensity in an infinitesimal solid angle $d\omega$ at the angle ϕ, from a unit volume having a unit cross section and a unit length."[McCartney, 1976]. The total scattering coefficient β however is the same for both polarised and unpolarised light as

described by equation 2.28.

$$\beta = \frac{128\pi^5 r^6 N}{3\lambda^4} \left(\frac{n^2 - 1}{n^2 + 2} \right)^2 \tag{2.28}$$

Since almost all parameters in this term are usually constant, the very elegant relation

$$\beta = c \cdot \lambda^{-4} \tag{2.29}$$

may be written, with c a constant accounting for all the parameters except λ. This best shows the essence of the Rayleigh law.

With this knowledge one can describe the phenomena of the blueness of the sky as well as the red colour of the sun. Because of the angular component of the Rayleigh scattering, which is stronger for short wavelengths, such as blue, the blue component of the sunlight gets scattered more and will be spread orthogonal from the source of light. This results in the illumination with blue light of the surrounding particles, which produces an overall diffuse blue sky light. In direction to the sun however, the air molecules scatter the long wavelength less, hence transmit long waves better in forward direction from the sun, letting the sun seem yellow/red, since the shorter wavelengths are missing in the forward direction. This phenomenon also affects how objects in the atmosphere appear to a distant observer, that is to say with a blue hue. This is true for clear air, which is why the sky appears even more blue when the air is very clear. During sunrises and sunsets, the effects of Rayleigh scattering are much more noticeable because of the longer distance through the atmosphere that sunlight has to pass. When the air is contaminated by larger particles such as haze aerosols, the sky appears less blue and one must use the Mie scattering theory to describe the phenomena as it will be described in the following section.

2.3.4. Mie Scattering

There is a gradual transition from Rayleigh to Mie scattering as the particle size increases, whereas the scattering pattern of Rayleigh is symmetrical and comparatively

simple, the scattering pattern of Mie scatterers gets more complex with increasing particle sizes, as it is shown in figure 2.9 on page 29. Also with size increase, there is a noticeable increasing ratio of forward scattering to backscattering, which results in a growth of the forward lobe, also shown in figure 2.9. The dependency of scattering on wavelength, however, becomes much less significant. As may be inferred from the generally white appearance of clouds, since the water droplets in clouds exceed the size of Rayleigh scatterers by several orders. The present theory has been developed by many workers, all based on the initial researches of Mie(1908), much of it was anticipated by Lorentz in the period 1890 to 1900 which is why the theory is sometimes called Mie-Lorentz theory. Likewise as for the Rayleigh theory a spherical scatterer is assumed. Non spherical particles as described by Dessens and shown in figure 2.4 on page 19 would be very complex and depend on the angle of the incident light to the particle orientation in space. Non spherical particles are therefore not in the scope of this thesis, when considering water droplets in liquid state, they are approximately spherical anyway. Mie scattering by a single oscillator is closely related to the scattering principle introduced for Rayleigh scattering and thus based on electromagnetic waves interacting with a dipole oscillator. Although the physical basis has been shown already in the Rayleigh scattering section, the Mie theory is more complex in detail, thus the reader may be guided directly to the scattering functions for mono dispersions of spherical oscillators. Because of the many closely packed particles in the atmosphere, interferences among the scattered waves occur as a function of observation angle. The angular scattering function at observation angle θ for unpolarised light can be written as:

$$I(\theta) = \frac{I_{\perp}(\theta) + I_{||}(\theta)}{2} = E\frac{\lambda^2}{4\pi^2}\left(\frac{i_1(\theta) + i_2(\theta)}{2}\right) \tag{2.30}$$

where $I_{||}$ and $I_{\perp}$ are the two components of the scattered light, polarised perpendicular and parallel to the plane of observation, respectively. The two components are proportional to the intensity distribution functions i_1 and i_2, respectively. These functions are the essence of the Mie theory and depend on the size ratio defined by 2.22, on the refractive index m and the observation angle θ. These functions are expressed by

$$i_1(\alpha, m, \theta) = |S_1|^2 = \left| \sum_{n=1}^{\infty} \frac{2n+1}{n(n+1)} (a_n \pi_n(\theta) + b_n \tau_n(\theta)) \right|^2 = |Re(S_1) + Im(S_1)|^2 \tag{2.31}$$

and

$$i_2(\alpha, m, \theta) = |S_2|^2 = \left| \sum_{n=1}^{\infty} \frac{2n+1}{n(n+1)} (a_n \tau_n(\theta) + b_n \pi_n(\theta)) \right|^2 = |Re(S_2) + Im(S_2)|^2 \tag{2.32}$$

Where n's are positive integers and the values a_n and b_n are found from Ricatti-Bessel functions, whose arguments are formed from the particle characteristics α and m but which are independent from the angle θ. The functions π_n and τ_n depend only on the angle θ and involve the first and second derivatives of the Legendre polynomials with order n and argument $cos(\theta)$. In these functions, the refractive index that is denoted by m, in contrast to the commonly used n, expresses the influence of absorption, it may be written as:

$$m(\lambda) = n(\lambda) - i \cdot n_i(\lambda) \tag{2.33}$$

where n and n_i now are the real and imaginary parts, respectively. The imaginary part, which is identical to the *absorption index* k, is related to the Lambert absorption coefficient K:

$$k \equiv n_i = \frac{K\lambda}{4\pi}. \tag{2.34}$$

When $\alpha \ll 1$ and $m \approx 1$, the first term in each series degenerates to the Rayleigh scattering function. The angular scattering function depends greatly on the polarisation of the primary wave. A scattering function for each type of polarisation, perpendicular, parallel or otherwise polarised light can be written down, but since common light sources such as sunlight are unpolarised, only such will be considered throughout this section. It may be noted that the calculations from equations 2.31 and 2.32 are obviously very

impractical in manual use and also comparatively slow on computers.

Similar to the extinction coefficients defined for the Rayleigh theory, one can define scattering coefficients for the Mie theory. The volume total scattering coefficient is defined as

$$\beta_{sc} = N\pi r^2 Q_{sc} \tag{2.35}$$

and likewise for the absorption and extinction, respectively:

$$\beta_{ab} = N\pi r^2 Q_{ab} \tag{2.36}$$

$$\beta_{ex} = N\pi r^2 Q_{ex} \tag{2.37}$$

with N the number of particles and with Q being the efficiency factor, or often called the *Mie coefficient* K. The efficiency factor defines how much a particle scatters the incident light. Q may be written as

$$Q_{sc} = \frac{1}{\alpha^2} \int_0^{\pi} (i_1 + i_2) \sin\theta \; d\theta \tag{2.38}$$

Q has the same dependence on the refractive index as the intensity functions i_1 and i_2. And obviously a dependence on the size parameter α, both in the intensity functions and in the denominator. The extinction stays in relation to scattering and absorption in a way that:

$$Q_{ex} = Q_{sc} + Q_{ab} \tag{2.39}$$

Now the angular scattering coefficient $\beta(\theta)$ may be written, referring to equation 2.30, as:

$$\beta(\theta) = N\frac{\lambda^2}{4\pi^2}\left(\frac{i_1 + i_2}{2}\right) \tag{2.40}$$

When now considering a polydispersion, in the atmosphere droplet size distributions may vary over several orders of magnitudes, however, the Mie theory is valid even for those extreme size distributions. In applying the preceding theory to polydispersion it is necessary to integrate over the size distribution. In the first step, equation 2.40 becomes:

$$\beta_t(\theta) = N\frac{\lambda^2}{4\pi^2} i_t(r, \lambda, m, \theta) \tag{2.41}$$

with $i_t = \frac{i_1+1_2}{2}$. Equation 2.41 is now extended to polydispersion by substituting N by the power law distribution function 2.8(from page 22):

$$\beta(\theta) = 0.434c\int_{r_1}^{r_2} \frac{\lambda^2}{4\pi}\frac{i_t}{r^{v+1}}dr. \tag{2.42}$$

Similarly the volume total scattering coefficient for the power-law size distribution function is expressed by

$$\beta_{sc} = 0.434c\pi\int_{r_1}^{r_2} Q_{sc}\ r^2\frac{1}{r^{v+1}}dr. \tag{2.43}$$

To illustrate the angular volume scattering, the reader may refer to the figures 2.13(a-f). The figures show the scattering pattern with altered parameters, a-c show alterations of the sphere radius from 0.025 to 0.25μm both on a linear and a logarithmic scale with a wavelength of 555nm. The figures d-f show the dependency on wavelength of a scatterer being 0.25μm in radius over the wavelength from 200nm(ultraviolet) to 800nm(infrared). It was assumed a concentration of $10^3 cm^{-3}$, which is about the concentration of haze particles in the atmosphere(compare table 2.1, page 15) and a refraction index of 1.0($Re = 1.0, Im = 0.0$). In all figures a-f, the light is incident from the left on a sphere located at the center of the polar plot. With the colours meaning, red: unpolarised, green: perpendicular polarised and blue: parallel polarised light.

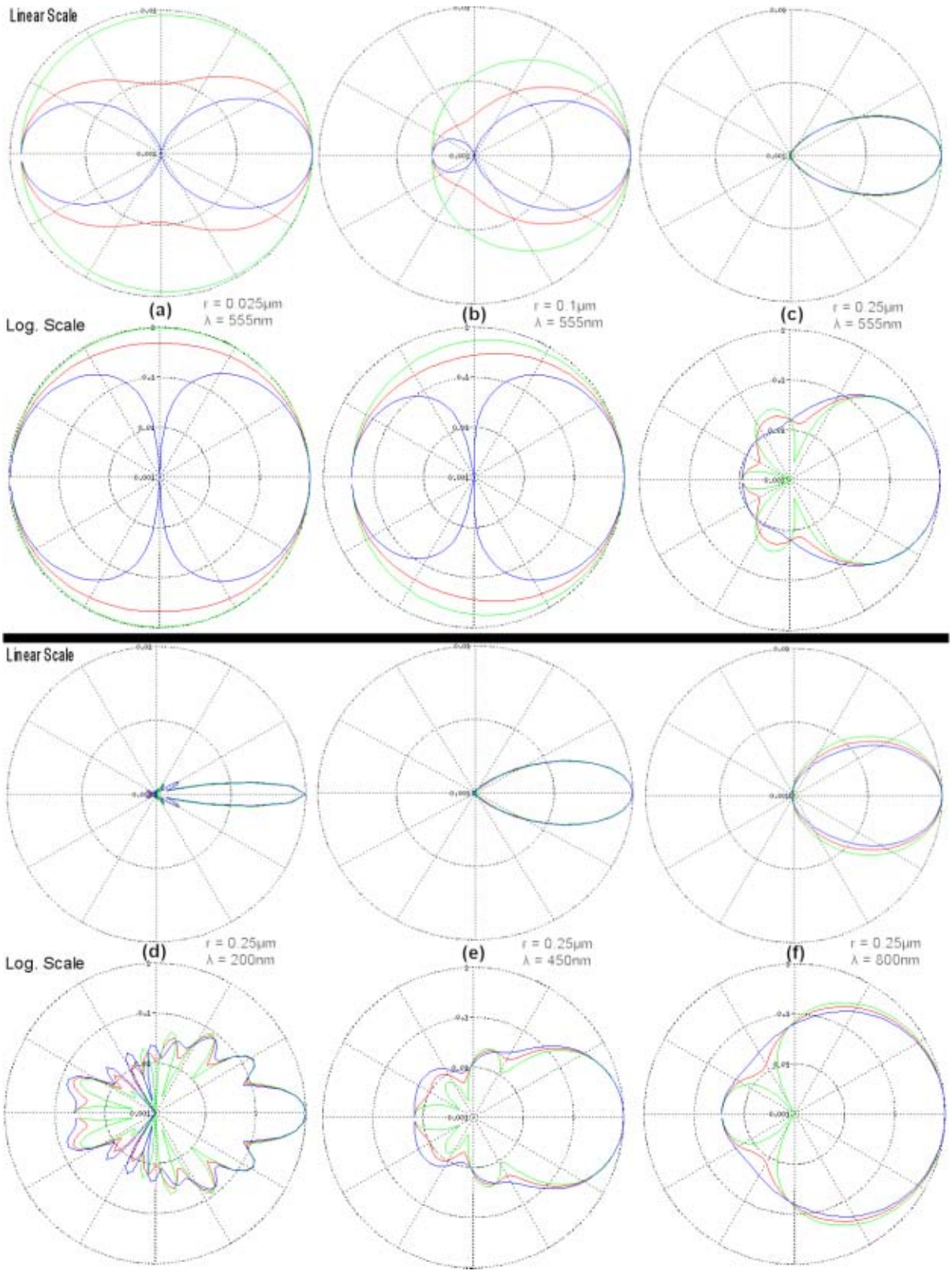


Figure 2.13.: Angular scattering intensity of a Mie scatterer in XY plane, figures were produced with [Online Mie Calculator, 2007]. Top two rows(figures a-c) show the alterations over particle size(top: linear scale, bottom: logarithmic scale), bottom two rows(figures d-f) show the alterations over wavelength. In all figures the light is incident from the left.

As one can see from figures 2.13a-c, with increasing particle size the forward scattering increasingly becomes the dominant form of scattering. With $r = 0.025\mu$m, the particle size is near an air molecule and the pattern degenerates to such of a Rayleigh scatterer, whereas with $r = 0.25\mu$m the scattering pattern is much more complex, as very apparent from figure 2.13c(logarithmic scale), as it would be for large haze particles or cloud droplets. From figure 2.13d-f one can see how the scattering pattern becomes less complex with increasing wavelength λ. As shown in the figures, a particle with $r = 0.25\mu$m which is equivalent to a haze particle, scatters light with longer wavelength more than that of shorter wavelength like ultraviolet and blue wave lengths, thus a camera with a front mounted light, observing a scene through haze(not in general, just with particle radius: 0.25μm!) would produce images with greater visual range when using ultraviolet light rather than infrared light. Irrespective of possible advantages of infrared sensors due to heat radiation of targets in the scene that might be especially desirable, such as warm airplanes on a cold runway.

Further dependencies on the wavelength apply for the attenuation. In general, the dependence can be described, at least for a narrow wavelength range, by

$$\beta_{sc} = constant \times \frac{1}{\lambda^{\gamma}}. \tag{2.44}$$

The exponent γ may vary from four, for Rayleigh scattering(very small particles), to practically zero for visual and near infrared wavelengths in fog. Between these two extremes lays the wavelength dependence of an actual haze. The relationship is more complicated than that, however. McCartney(1976) put it this way: "Any fog(or cloud) scatters selectively in one or another part of optical spectrum, depending on the values of α over the size distribution and the given spectral range, and on the relationship of Q_{ex} to α."[McCartney, 1976]. When $\alpha \ll 1$(Rayleigh scattering), the extinction coefficient becomes

$$\beta_{ex} = constant \times \frac{1}{\lambda^{\gamma \to 4}}. \tag{2.45}$$

This condition is realised at radar and extreme infrared wavelengths in fog and cloud. When the wavelength is shorter or at least comparable to the droplet sizes, the depen-

dence on λ fluctuates strongly. When $\alpha \gg 1$(Mie scattering), which is true for fog and visual wavelength, the scattering coefficient becomes

$$\beta_{sc} = constant \times \frac{1}{\lambda^{\gamma \to 0}} \tag{2.46}$$

which shows that the dependence is practically zero at these relatively short wavelength. This phenomenon is shown in figure 2.14, the relative attenuation of blue and red are flipped when going from a small scattering coefficients to larger scattering coefficients, at the transition between haze and fog, known as *mist*. Whereas green wavelengths are relatively constant over the entire scale of mean β_{sc}. It is Remarkable that beyond $\beta_{sc} = 6km^{-1}$ the relative attenuation remains stable and is no longer spectrally selective in the visual region as the fog thickens. Below that value of β_{sc}, which corresponds to haze, the attenuation becomes spectrally selective.

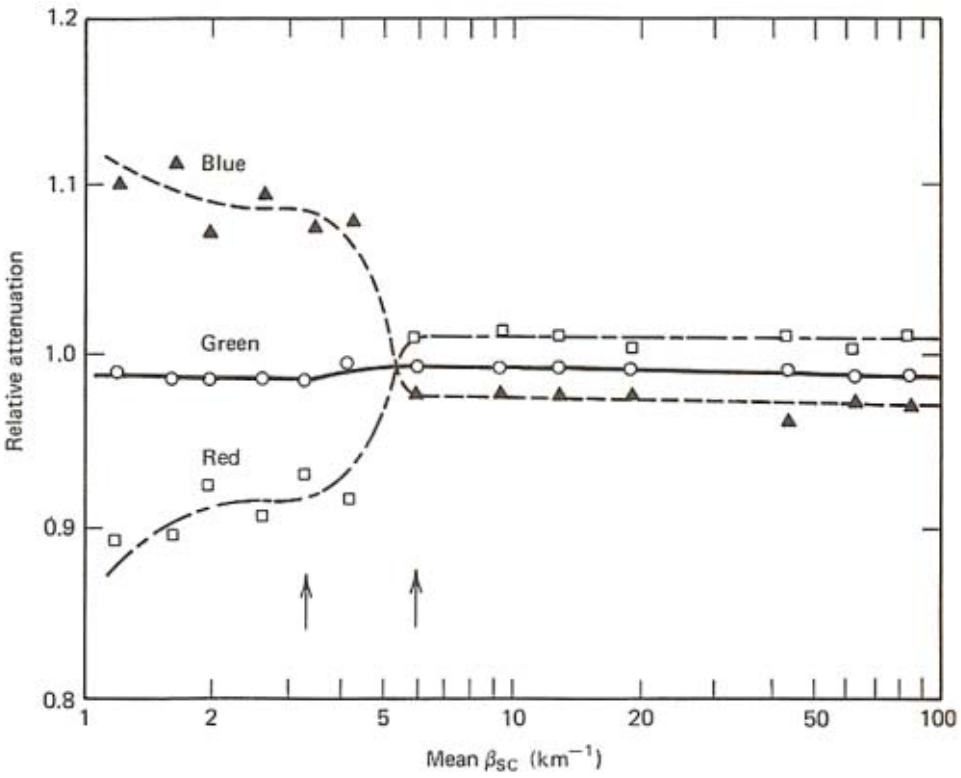


Figure 2.14.: Relative attenuation of blue, green and red light in fog. The arrows indicate the transition condition between haze and fog, also known as mist. On the ordinate are different states of haze thicknesses with constant distance. *Source:* [Eldridge, 1969] and [Foitzik, 1938].

It was often tried to find a relation likewise elegant to this of the Rayleigh law 2.29(page 36), then Granath and Hulbert(1929) proposed the equation:

$$\beta_\lambda = c_1\lambda^n + c_2\lambda^{-4} \tag{2.47}$$

which seems to do the trick for the spectral scattering coefficient. This is in agreement with what Dessens found later in 1947. Dessens, who investigated the particle sizes and their distributions(see page: 19), showed a remarkable correspondence between the spectral scattering coefficient, calculated from his theoretical calculations of size and size distribution after he collected particles with his spiderweb method, and those obtained directly from an optical measurement while the particles were being collected. He measured the scattering coefficients of the wavelengths between 0.35 and 0.85μ, after the Rayleigh scattering was subtracted, and found that the exponent n in equation 2.47 varies between 0.3 and 1.3. He also confirmed that particles with a radius $r < 0.2\mu$ have only a very small optical effect(Rayleigh Scattering) because of the smallness of r^2 and K, the scattering area ratio. With this experiment he found more evidence for the validity of the Mie theory, which is the most accepted theory today for light scattering by haze aerosols.

2.4. Koschmieder's Theory

In this section, the informations of the human eye and the atmosphere are put together to a holistic theory of horizontal visibility through the atmosphere. The first who was able to take the important step from the Mie scattering theory to the "Theorie der horizontalen Sichtweite" was H. Koschmieder in 1924. This chapter describes his theory in detail, beginning with a concept called the *air light*.

2.4.1. The Air-Light

Air light is an important concept, that is known to everyone, however not always by this name, who observed a natural or urban extensive landscape from an elevated point. The

observer will notice that those parts of the foreground that are closer to the horizon will be lighter in tone than those closer to the observer, this effect becomes more apparent the further away the objects are until they eventually blend into the horizon sky and become invisible. This way, even a black object becomes lighter and even appears in the same colour as the sky when it is seen far enough from the observer. The source for this increase in luminance with distance is of course the sunlight which is scattered into the eyes of the observer from the air, that is in the line of sight between him and the observed object. This phenomenon has been known for a long time and was used by painters for hundreds of years, the first quantitative analysis however was made by Koschmieder in 1924.

2.4.2. "Theorie der horizontalen Sichtweite"

Koschmieder published his "Theorie der horizontalen Sichtweite", which is German for theory of the horizontal visual range, in an extensive paper. The essentials of it are presented here. Koschmieder made some not too restrictive but simplifying assumptions that can be summarised as in the following enumeration (taken from [Middleton, 1952]):

1. "The atmosphere is considered as a turbid medium, containing a large number of small particles."(see the size distribution chapter, page 18).

2. "Each element of volume contains a very large number of particles, each of a much smaller order of magnitude than the element itself."(see the size of particles table, page 15)

3. "The scattering action of each particle is independent of the presence of all the others."(this also was initially assumed in the Mie scattering section, page 37)

4. "The light scattered from an element of volume will be considered as coming from a point source of which the intensity is proportional to the number of particles."(This is equivalent to the assumption that the light from the various particles is incoherent, which means that there is no stable phase relation. And since it was shown on page 27 that the average spacing is great enough so that the motions of the molecules are mostly independent, the assumption is approximately true as well.)

5. "Light rays will be considered as rectilinear, that is to say atmospheric refraction will be neglected."

6. "All parts of the sky are equally illuminated."(This assumption is the rephrased one by Middleton(1952) to make the initial unnecessary assumption less restrictive.)

Later, the earlier assumptions made by the physicist Gruner(1919) were found convenient for this theory and have thus been added:

7. "The coefficient of attenuation by scattering, β, is constant in a horizontal plane, in particular near the surface of the earth, where it takes the value β_0."(Despite the earlier argumentation, in practice this is approximately true in the atmosphere for a defined location, as particle size distribution is mostly a function of altitude and can therefore be assumed constant for a constant altitude.)

8. "The curvature of the earth is neglected, and its surface is considered as plane, horizontal, and diffusely reflecting."(Which is approximately the case for visual ranges anyway.)

9. "The linear dimensions of the whole observed object are small in comparison to its distance from the observer."

With image dehazing in mind, the seventh assumption implies that this theory can not be used for images taken from satellites or other aerial photography such as from a plain downwards. However it is possible to dehaze those types of images too, admittedly with slight alterations to the theory.

After Koschmieder made the initial assumptions, he integrated the illumination over a cone of air, illuminated by the diffuse sky light, the diffuse ground light and the sun directly, as shown in figure 2.15.

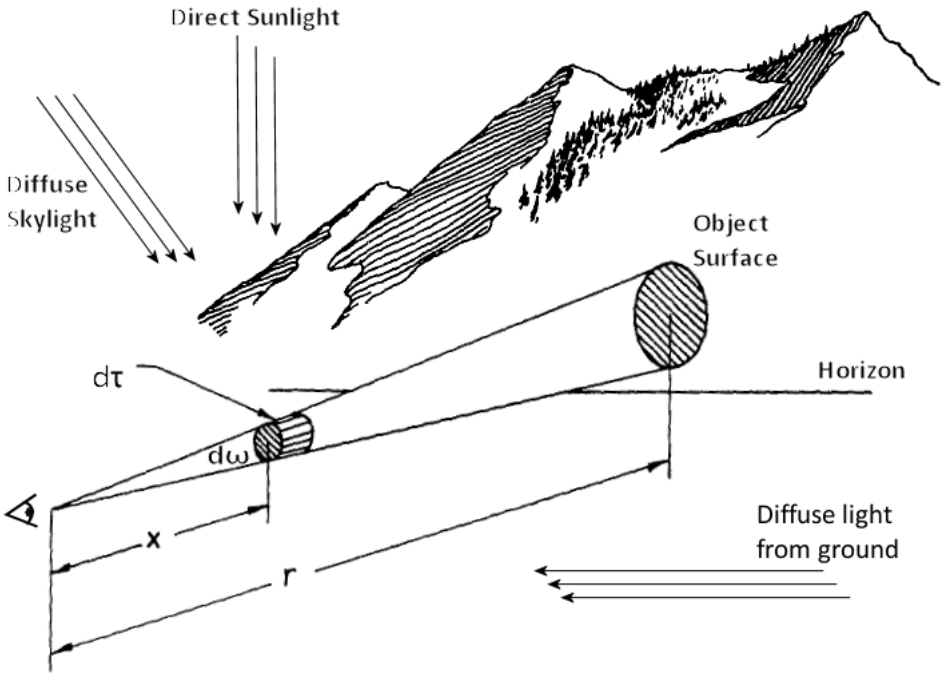


Figure 2.15.: Illustration for Koschmieder's theory, showing the volume element $d\tau$. Adapted from [Dickson et al., 1961]

Although it would exceed the boundaries of this thesis to follow him through the entire integrations, the most important steps are summarised. He considered an element:

$$d\tau = d\omega \cdot x^2 dx \tag{2.48}$$

for the cone of air, as in figure 2.15 and the object being a black object rising above the horizon at distance r. The observer being at $x = 0$. Now for any given value of ϕ the scattering function $\beta(\phi)$ will be proportional to the scattering coefficient β (see equation 2.21, page 30), therefore for the particular distribution of incident illumination, the intensity of the volume element $d\tau$ in the direction of the eye will be

$$dI = d\tau \cdot A\beta \tag{2.49}$$

With A a constant of proportionality, to be determined from boundary conditions, as it will be shown next. According to Koschmieder, the illuminance L of the scene at the eye due to the scattered light by the air from $d\tau$ is

$$dL = dI \cdot x^{-2} \cdot e^{-\beta x} \tag{2.50}$$

Here $e^{-\beta x}$ accounts for the transmittance of the thickness x of the atmosphere. When considering the solid angle $d\omega$ within the volume element $d\tau$, the equation becomes

$$dL = \frac{1}{d\omega} dI \cdot x^{-2} e^{-\beta x} \tag{2.51}$$

or writing dI in terms of x and $d\omega$, it is

$$dL = A\beta e^{-\beta x} dx \tag{2.52}$$

which can be integrated over the entire cone to get the apparent luminance of the object

$$L_\beta = \int^r A\beta e^{-\beta x} dx = A(1 - e^{-\beta r}) \tag{2.53}$$

This is the A as it was referred to earlier. Supposing a black object is infinitely far away, then the value of its luminance will converge to the luminance value of the horizon sky L_h. Thus

$$L_h = \int_0^\infty A\beta e^{-\beta x} dx = A \tag{2.54}$$

and therefore

$$L_\beta = L_h(1 - e^{-\beta r}) \tag{2.55}$$

This is probably the most important formulation in Koschmieder's theory, however this is only true for a black object, that is to say, none of the luminance actually originated at the object itself. This condition is rather impractical, fortunately with a slight alteration, the formula can account for an object having a luminance L_0.

$$L = L_0 e^{-\beta r} + L_h(1 - e^{-\beta r}) \tag{2.56}$$

With equation 2.56 the theory is very close to a form which can be used directly in applications, however not quite there, since it does not take into account the absorption by the atmosphere. Because absorption is often of comparable magnitude to scattering, the extinction coefficient(equation 2.16, page 30) is used in the following formula:

$$L_\beta = L_0 e^{-(\beta+k)r} + L_h(1 - e^{-(\beta+k)r}) = L_0 e^{-\sigma r} + L_h(1 - e^{-\sigma r}) \tag{2.57}$$

As a side note, when using $\beta + k$ it is also called the "two-constant theory".

2.4.3. Consequences of Koschmieder's Theory

As it was shown in section 2.1(5), the decrease in contrast directly causes a lower visibility. This is now, thanks to Koschmieder, quantifiable. Let L_0 and L_0' the luminances of two objects at close distance, then it is at distance R:

$$L_R - L_R' = (L_0 - L_0')e^{-\sigma_0 R} \tag{2.58}$$

now using the contrast, as defined by 2.1(page 10)

$$C_0 = \frac{L_0 - L_0'}{L_0'} \tag{2.59}$$

which is the *inherent contrast*, and

$$C_R = \frac{L_R - L'_R}{L'_R} \tag{2.60}$$

that is to say the *apparent contrast*. Now equation 2.58 may be written as

$$C_R = C_0(L'_0/L'_R)e^{-\sigma_0 R} \tag{2.61}$$

This formula is the law of contrast reduction as it was discovered by Duntley(1948). Koschmieder's theory has been verified by many field studies to contrast and visual range since its development. It is the most important basis for visual range calculation and of course dehazing methods today. As the reader will see in the next chapter, it is used among the entire variety of model based dehazing algorithms.

3. Dehazing Methods

Many dehazing algorithms have been developed by different workers that were very active in this field of study in recent years. This chapter will describe the theoretical principles and main differences between these methods. Throughout this chapter the reader should keep in mind, that the author searches for a dehazing method that can be used for a real-time application, thus the methods will be analysed towards these aspects.

3.1. Overview of Dehazing Methods

To achieve the goal of haze removal and visibility improvement, several ways can lead to decent results. The methods can basically be divided into two groups, those methods that only need one single image for dehazing and those methods using 2 or more input images for one dehazed image. Also, there are model based methods and those trying to enhance the contrast of an image using simpler computer vision techniques such as gamma correction, unsharp masking or histogram equalisation. The model based methods usually produce a depth map of the scene as a byproduct, the variety of applications that could make use of the depth information seems to be endless, Kopf et al.(2008) explored some possibilities in their paper. A video presenting these results is worth seeing for the interested reader(`http://johanneskopf.de/publications/deep_photo/index.html`). The question of what method gives the best visibility improvement should best be answered by the human, as he will be the ultimate judge using the dehazed image in applications such as surveillance scenarios. It should therefore be evaluated in psychological studies, testing the probability of detection and employing measurements of tiredness of the eyes using the one or the other dehazing method. A comparison of a broad variety of methods can be seen in figure 3.1, the images are taken from [Fattal, 2008].

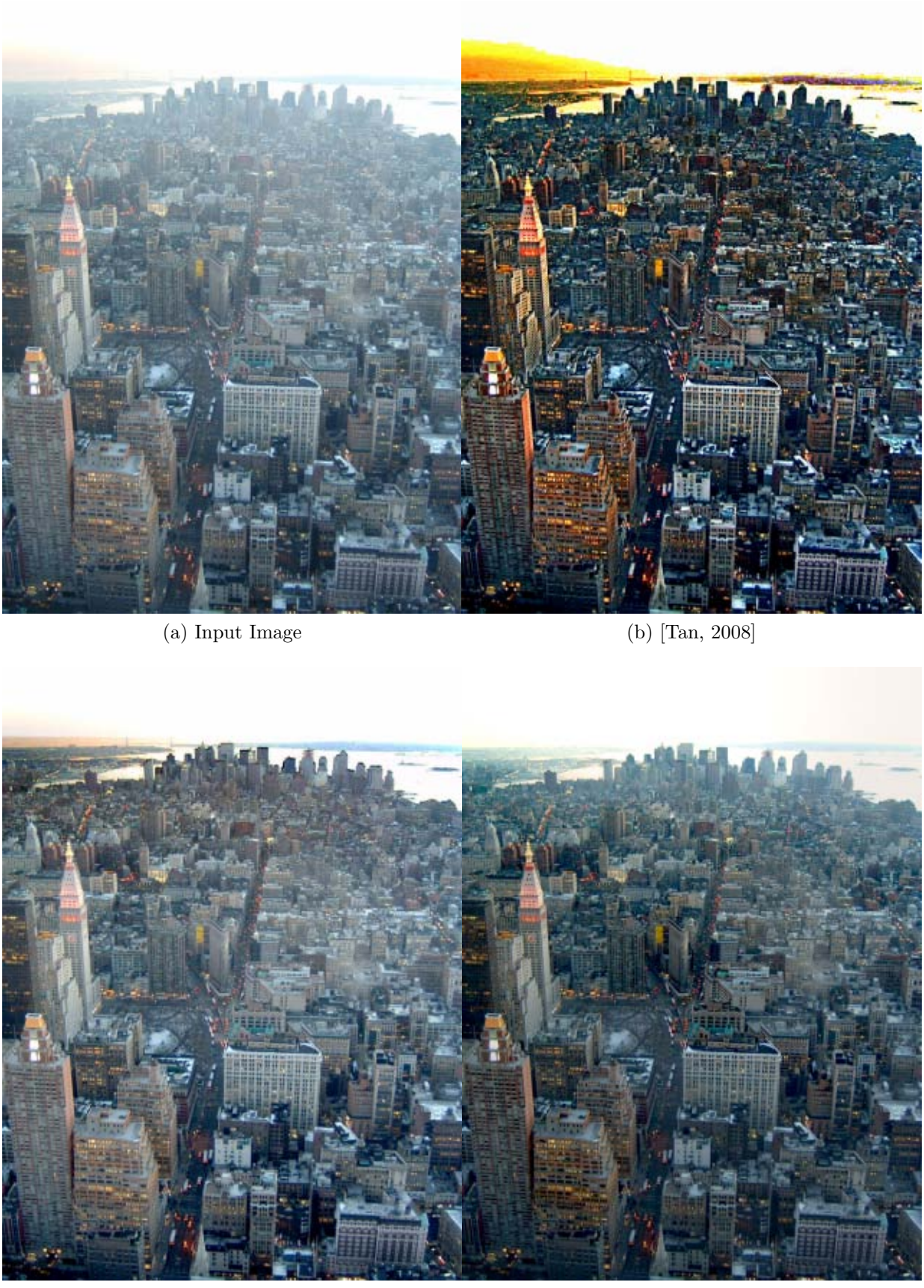
(a) Input Image (b) Plarisation-Based [Schechner, et. al., 2001]

(c) Gamma Correction (d) Histogram Equalisation (e) Unsharp Masking

(f) [Fattal, 2008] (g) Dark Channel Subtraction, [Chavez, 1988]

Figure 3.1.: Comparison of computed images from the same input image created with different dehazing techniques. *Source:* [Fattal, 2008]

Figure 3.1(a) shows the input image, right next to it in 3.1(b) is the result of a polarisation based dehazer, due to its outstanding image quality this method can be seen as reference method. The next row 3.1(c)-(e) contains images, whose visibility have been improved by simpler image processing methods such as gamma correction, histogram equalisation and unsharp masking, respectively. The last row contains the results of Fattal(2008) and Chavez(1988), Fattal's results are the best for the single image dehazing methods in this comparison, which shows that single image model based, recently developed methods are the way to go. As of three years prior the time of writing this thesis, a couple of works with outstanding results have been published. He et al.(2010) have published a new method based on the dark channel subtraction from Chavez(1988), that produces comparable results to those of Fattal. Also Tan(2008) and Kopf et al.(2008), respectively, are workers in this field with their own image dehazing methods each. The latter gives outstanding results, and is one of the best in quality available today. These four works are compared in the figure 3.2 and 3.3 , the images are from [He et al., 2010b].



(a) Input Image (b) [Tan, 2008]

(c) [Kopf et al., 2008] (d) [Fattal, 2008]

Figure 3.2.: Comparison of computed images from the same input image created with different model based single image dehazing techniques. *Source:* [He et al., 2010b]

(a) [He et al., 2010a] (b) [He et al., 2010a]

Figure 3.3.: The figure(a) shows dehazing result using the algorithm of [He et al., 2010a], using 3.2a as input image. Figure (b) shows the depth map of the scene computed using the same algorithm. *Source:* [He et al., 2010b]

It is worth mentioning, that all dehazing methods no matter what principle they use, depend somewhat on the dynamic range of the image sensor itself. This is a limiting factor, less in CCDs and more in CMOS sensors but present in all todays available image sensors. Image quality, especially the contrast could be enhanced when using higher dynamic range sensors [Namer and Schechner, 2005].

It will now be drawn attention to each one of the introduced methods in order to find out the method with the most potential for real-time applications without sacrificing too much of the image quality of the respective dehazing method.

3.2. Non-Model-Based Contrast Enhancer

The middle row in figure 3.1 shows the results of the non model based contrast enhancers. The striking similarity is the blue hue that the three have in common, this is the typical blueness of small haze particles as described in the preceding chapter. This can only be eliminated with model based techniques. These are just a few examples, there are actually many contrast manipulation algorithms available, mostly known from photography. As the reader saw from the introduced principles of contrast(equation 2.2, page 11), humans have a minimum contrast threshold that is needed for object separation. Luckily, this contrast threshold can be raised in images using simple mathematical concepts like the gamma correction, unsharp masking or histogram equalisation. These were not developed to dehaze images, can however improve visibility for the human eye, hence they can also be used to further improve an already dehazed image. Thus for this purpose, they shouldn't be used exclusively, but in combination with a dehazer. For the sake of completeness however, the three introduced concepts will now be described very briefly.

3.2.1. Unsharp Masking

The idea behind unsharp masking is to emphasise edges. An unsharp masking algorithm can detect edges and alter the levels of brightness on both sides of the edge in a way that the darker side gets even darker towards the edge and the lighter parts get even lighter towards the edge [cambridgeincolour.com, 2011]. This produces an overshoot and undershoot, respectively of the brightness curve of the pixels at the edge. This is illustrated by figure 3.4. An unsharp mask amplifies high-frequency components. This results in a better sharpness of the image, thus raising the contrast locally. The ideal

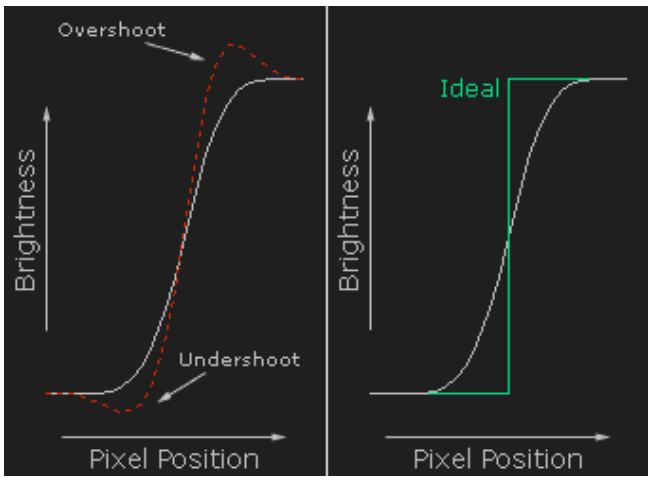


Figure 3.4.: Shows the principle of unsharp masking, left plot shows the overshoot and undershoot resulting from the masking versus the actual brightness curve, right plot shows the ideal rectangular curve versus the actual curve of the input image. *Source:* [cambridgeincolour.com, 2011]

brightness curve would be a step function,
as illustrated by the right plot in figure 3.4.

3.2.2. Gamma Correction

Gamma correction refers to a nonlinear operation that amplifies or reduces the luminance intensity of an image. This operation is performed on each pixel in the same way, no matter its original value. By lowering γ in the power-law expression for the gamma correction:

$$B_{out} = B_{in}^{\gamma} \tag{3.1}$$

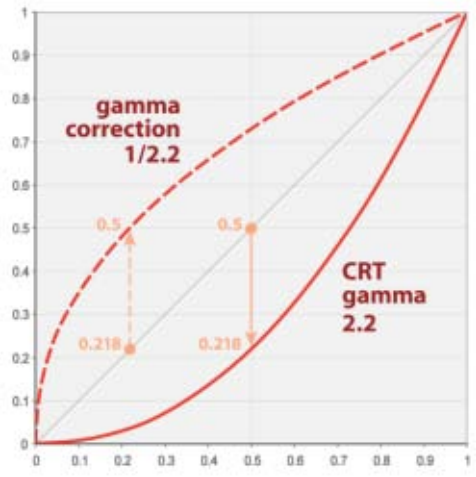


Figure 3.5.: Plot shows the principle of gamma correction, *Source:* [wikipedia.org, 2011b]

the contrast levels may be raised in dark images with low contrasts. Here B_{in} and B_{out}, respectively denote the Brightness levels before and after the gamma correction. The principle of gamma correction can be seen in the plot of figure 3.5, the input brightness curve is here as an example the input of a CRT display with $\gamma = 2.2$, with a gamma correction curve of $\gamma = 1/2.2$ the actual linear curve of interest can be restored. The ordinate shows the input value(brightness) and the abscissa shows the output value(brightness).

3.2.3. Histogram Equalisation

The histogram equalisation is a method using the image's histogram in order to improve contrast. Since only so much of brightness values can be displayed, all gradations of brightnesses and therefore contrasts must be within the brightness bandwidth. Thus the best results can be obtained, when spreading out the most frequent intensity values over the entire histogram.

A disadvantage of this method is that it may increase the noise by discriminating it from the actual usable signal. However, this method is one of the more advanced methods to improve image contrast, and is of the three mentioned in this thesis the most resource intensive, but usually also the one with the best results. Figure 3.6 illustrates the basic idea behind histogram equalisation. The left plot shows the input, where not all possible brightness values are used and the right plot after the transmission. This method generally improves the global contrast, locally however spots in the image with close brightness values may not be improved in some cases.

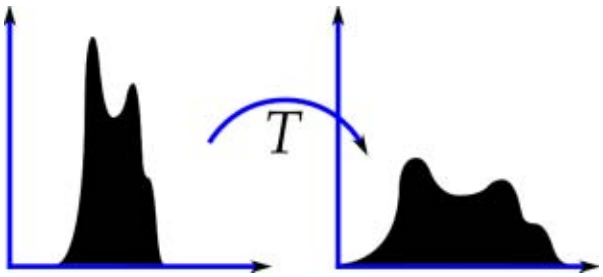


Figure 3.6.: Plot shows the principle of histogram equalisation, the abscissa denotes the brightness values. The ordinate denotes the frequency that x-value is used. *Source:* [wikipedia.org, 2011c]

3.3. Polarisation Based Visibility Improvements

Polarisation-based dehazing methods are part of the multi-image group, they usually use two input images taken with two differently polarised filters, one after another, to produce one dehazed image. This technique makes use of the fact that the airlight is at least partially polarised(see section 2.3, page 27), whereas the direct transmission of the object is unpolarised. Polarisation filters alone cannot eliminate haze in scenes, at least two images with different polarisation filter states are necessary. The assumed model is that of Koschmieder's theory with additional polarisation filters, see figure 3.7.

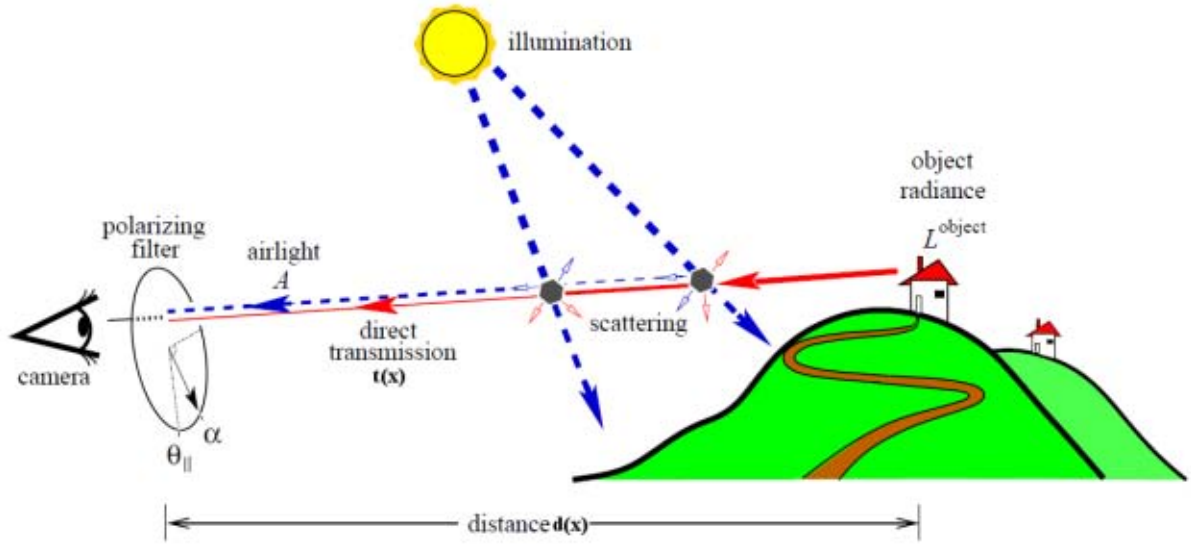


Figure 3.7.: Model for polarisation-based dehazing. *Source:* [Namer and Schechner, 2005]

With the knowledge from the preceding chapter, this figure is pretty much self-explanatory. In this section, the basic principles ae described behind the method shown in [Namer and Schechner, 2005], however this is also representative for all other known polarisation-based methods.

Consider figure 3.7, the resulting image of a scene consists of two main components, the first coming from the object radiance L^{object} and the second from the airlight A. The former is free of scattering in the line of sight and is only dependent on the attenuation of the atmosphere. The direct transmission $t(x)$ can then be written as:

$$t(x) = L^{object} \cdot \tau \tag{3.2}$$

where

$$\tau = e^{-\beta d(x)} \tag{3.3}$$

is the transmittance of the atmosphere. Here $d(x)$ denotes the distance from the camera to the object and β the attenuation coefficient. The airlight, also called the *path radiance*,

is produced by the scene illumination and given by:

$$A = A_{\infty}(1 - \tau) \tag{3.4}$$

where A_{∞} is the saturation airlight, which depends on the atmospheric and illumination conditions. It is the maximal possible intensity of airlight, which corresponds to the airlight of the sky near the horizon. In contrast to the direct transmission the airlight factor increases with distance and dominates the image irradiance I_{total} at long ranges:

$$I_{total} = t(x) + A. \tag{3.5}$$

This is the major cause for reduction of image contrast in haze. The partially polarised airlight can now be used to restore a haze free image by mounting a polarisation filter with angle α in the imaging system. When rotating the polariser, there is an orientation at which the image is least intense, let this be denoted by I_{min}. I_{min} corresponds to the lowest amount of airlight. This orientation of the polarisation filter may be denoted by $\theta_{\parallel}$. When now rotating the pol. filter by 90° relative to $\theta_{\parallel}$, then the image irradiance is strongest, since now the principle polarisation component of the airlight is strongest, this may be called I_{max} with $\theta_{\perp}$. Once these two images are acquired, the dehazed image can be estimated by:

$$\widehat{L}^{object} = \frac{I_{total} - \widehat{A}}{\widehat{\tau}} \tag{3.6}$$

where the estimated transmittance, $\widehat{\tau}$ is:

$$\widehat{\tau} = 1 - \frac{\widehat{A}}{A_{\infty}} \tag{3.7}$$

and the estimated airlight $\widehat{A}$ is:

$$\widehat{A} = \frac{I_{max} - I_{min}}{p}. \tag{3.8}$$

Here p is the degree of polarisation of airlight, which depends on the particle size of the aerosols. It can however be estimated with just the parameters already known. It is measured from the raw images by looking at pixels which correspond to objects at infinity, naturally such pixels are those of the sky near the horizon(in later years, Schechner et al. proposed ways to find p without having a horizon in the picture).

$$p = \left.\frac{I_{max} - I_{min}}{I_{max} + I_{min}}\right|_{d(x)=\infty} \tag{3.9}$$

Hence the airlight saturation value can be estimated from the same sky area as

$$A_\infty = \left[I_{max} + I_{min}\right]_{d(x)=\infty}. \tag{3.10}$$

As stated in section 3.1, the distance map can be recovered as a byproduct:

$$\beta\widehat{d}(x) = -\log\left[1 - \frac{\widehat{A}}{A_\infty}\right]. \tag{3.11}$$

This operation must be done for each color channel separately. The method works with only slight alterations for both atmospheric photography [Schechner, et. al., 2003] as well as for underwater photography [Schechner and Karpel, 2004]. Problematic however, are surfaces that are specular, like windows or water, because they reflect partially polarised light and lead to errors in the calculation. These areas must be treated separately, and can for example be detected by analysing inconsistencies in the depth map, see [Namer and Schechner, 2005].

Although this method is model based, no knowledge about the actual scattering particles is necessary. However, due to its basic principle, only Rayleigh scattering can properly be eliminated. Since Mie scatterers polarise the light in a different way if any(the larger the particle, the less the polarisation). Also clear day recording is preferred for this

method, since then the airlight polarisation can properly be separated from the direct transmission. However, when the prerequisites are met, polarisation-based dehazing gives very good results, an example is given in figure 3.8. Note that the correct colours can also be restored in contrast to the formerly mentioned enhancers that purely focus on the contrast without being model based.

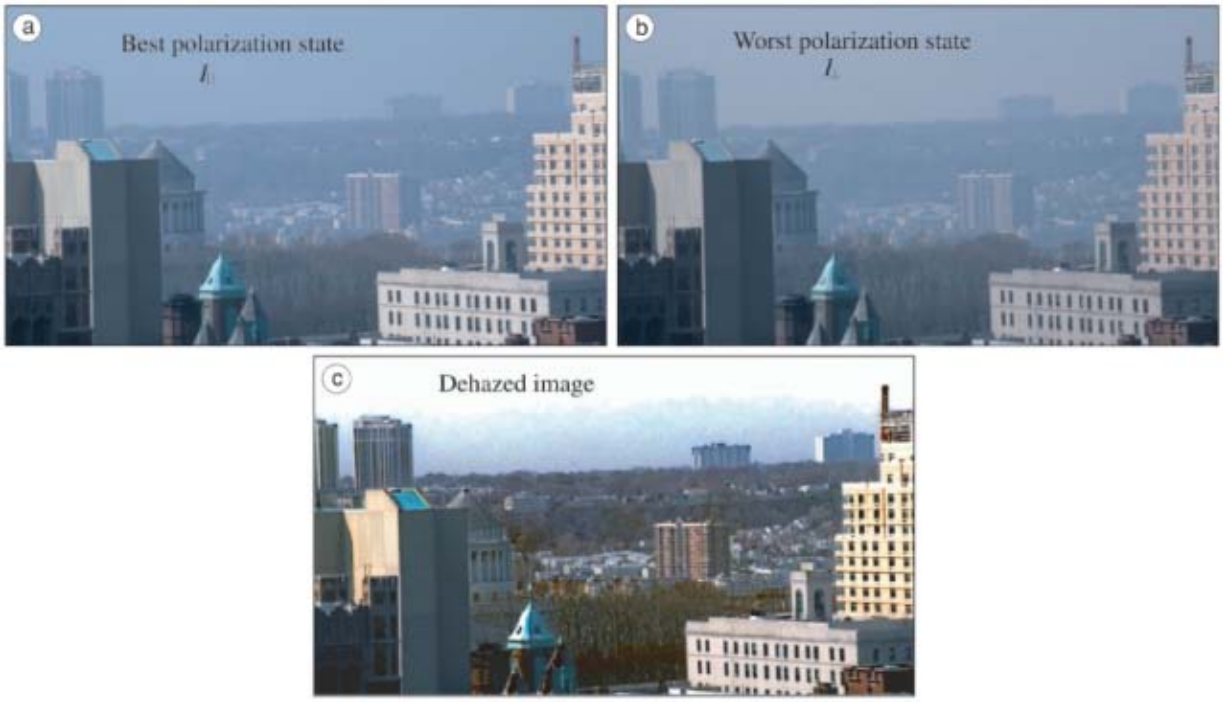


Figure 3.8.: Example for polarisation-based dehazing, (a) shows the image taken with a polarisation filter under best polarisation state $\theta_{\parallel}$, (b) shows the image taken with a polarisation filter under worst polarisation state $\theta_{\perp}$ and (c) shows the dehazed image. *Source:* [Schechner, et. al., 2003]

Multi image dehazing methods in general, although usually giving qualitatively great results tend to be unsuitable for real-time image dehazing purposes. It is thinkable that for example, half the framerate could be sacrificed in order to capture two images with different polarisation state, however rotating a static polarisation filter is of course very slow when done by hand or even done by a clocked mechanical mechanism. Although there are now electronic liquid crystal based polarisation filters available on the market that may be able to switch on every frame, they are very colour selective and give different results for different wavelengths of light [Namer and Schechner, 2005]. Namer and Schechner also stated, that: "While we still do not demonstrate dehazing in video, we currently assess the LC technology using still photography." [Namer and Schechner, 2005]. Also there

is the argument, that special hardware requirements may not be feasible in real world applications, since it may be inconvenient in practice to equip once deployed cameras with a liquid crystal polarisation filter for simple physical or monetary reasons. Especially in windy weather conditions, cameras that are mounted on high masts and equipped with a polarisation filter may be very sensitive to translation. Thus two images of different polarisation state may be useless when shifted by a few pixels or even subpixels due to wind or other causes(e.g. engine turbines of large air crafts causing low frequent vibrations in airport surveillance environments). Additionally to multi-image techniques, there are methods using multiple images such as the dehazing method of Nayar and Narasimhan described in [Nayar and Narasimhan, 1999] and [Narasimhan and Nayar, 2002], which require images taken under two different weather conditions. This is of course way too slow for real-time applications.

3.4. Fattal's Method

Fattal introduced a new technique in 2008 for single image dehazing that produces qualitatively great results on hazy images. The main idea of this approach is to take the image degradation model from Rossum and Nieuwenhuizen([Rossum and Nieuwenhuizen, 1999]) also known as the *Radiative Transport Equation*(shown in 3.12) and express it in terms of surface shading in addition to the transmission. This gives a refined image formation model. Quoting Fattal: "This allows us to resolve ambiguities in the data by searching for a solution in which the resulting shading and transmission functions are locally statistically uncorrelated. A similar principle is used to estimate the color of the haze." [Fattal, 2008].

$$I(x) = t(x)J(x) + (1 - t(x))A \tag{3.12}$$

In this equation, $t(x)$ is the transmission, a scalar for each colour component:

$$t(x) = e^{-\beta d(x)}. \tag{3.13}$$

Similar to the preceeding section the term $t(x)J(x)$ is being called the direct attenuation and $(1-t(x))A$ the airlight. Here $I(x)$ is the input image, $J(x)$ the haze free image and A the global atmospheric light colour vector. This equation is commonly used to describe the image formation in the presence of haze and was used before by, for example [Chavez, 1988],
[Nayar and Narasimhan, 1999] and [Shwartz et al., 2006]. This inherits many ambiguities in each pixel independently, such as in the airlight-albedo, that gives a large degree of freedom. Fattal however, manages to reduce this degree: "To reduce the amount of this indeterminateness, we simplify the image locally by relating nearby pixels together." [Fattal, 2008]. He does that by grouping pixels belonging to the same surface, thus having the same surface reflectance and therefore the same constant surface albedo. Now the key idea to resolve the airlight-albedo ambiguity is that he assumes that the surface shading l and the scene transmission t are statistically uncorrelated, because l depends on the illumination on the scene, surface reflectance properties and the scene geometry, whereas t depends on the density of the haze(β) and the scene depth. Fattal then presents an independent component analysis method to determine l and t. The same principle of uncorrelation is applied to the estimation of the airlight colour. This method also gives a depth map, which could be used over and over again for a static camera when using it in a real-time application. "The Method works quite well for haze, but has difficulty with scenes involving fog, as the magnitude of the surface reflectance is much smaller than that of the airlight when the fog is suitably thick." [Carr and Hartley, 2009]. According to Fattal, the noise level in the input image influences the quality of the dehazed image greatly. However, with Fattal's method the absolute error in transmission and haze-free image are both less than 7% in tests where the real haze free image was known, explanatory: "In its essence this method solves a non-linear inverse problem and therefore its performance greatly depends on the quality of the input data." [Fattal, 2008]. "Moreover, as the statistics is based on color information, it is invalid for grayscale images and difficult to handle dense haze which is often colorless and prone to noise." [He et al., 2010a]. For examples of performance, please refer to figure 3.1 and figure 3.2.

3.5. Tan's Method

Tan presented a single image based dehazing method in 2008, too. His proposed method is based on the optical model:

$$I(x) = L_\infty \rho(x) e^{-\beta d(x)} + L_\infty (1 - e^{-\beta d(x)}) \tag{3.14}$$

with L_∞ being the atmospheric light and $\rho(x)$ the reflectance, this formula is very similar to Koschmieder's equation 2.56(on page 49). The first term in this equation is the direction attenuation and the second term corresponds to the airlight A. He then expresses it in terms of light chromaticity and as a vector for the colour components. In this formula are more unknowns than knowns. Nevertheless, there are some clues or observations that Tan makes use of in his algorithm:

1. "The output image, [..] must have better contrast compared to the input image **I**." [Tan, 2008]

2. "The variation of the values of A[atmospheric light for the colour components] is dependent solely on the depth of the objects, *d*, implying that objects with the same depth will have the same value of A, regardless their reflectance(ρ). Thus, the values of A for neighbouring pixels tend to be the same. Moreover, in many situations A changes smoothly across small local areas. Exception is for pixels at depth discontinuities, whose number is relatively small." [Tan, 2008]

3. "The input images that are plagued by bad weather are normally taken from outdoor natural scenes. Therefore, the correct values of [the direct attenuation] must follow the characteristics of clear-day natural images." [Tan, 2008]

The author of this paper([Tan, 2008]) then proposes an algorithm employing the clues above. With **I** being the input image, the algorithm is:

1. Estimate L_∞

2. Compute α(light chromaticity) from L_∞

3. Remove the illumination colour of **I**

4. Compute the data term $\phi(p_x|A_x)$ from **I'**

5. Compute the smoothness term $\psi(A_x, A_y)$

6. Do the interference, which yields the airlight, A

7. Return the direct attenuation, $D\gamma'$, computed from A.

He also proposed a data cost function for step 4 in the framework of Markov random fields, which can be efficiently optimised by various techniques, such as graph-cuts. This algorithm is applicable for both colour and gray images. However, it does not recover the scene's original colour [Tan, 2008]. Despite its neat approach, this method is not easily applicable to real-time applications since it takes "The computational time for 600x400 images, using double processors of Pentium 4 and 1 GB memory, approximately five to seven minutes (applying graph-cuts with multiple labels)" [Tan, 2008]. Also, this method has some flaws compared to other methods of for example Fattal and Kopf et al. in terms of image quality. Since it produces halos near depth discontinuities and "The method tends to produce over enhanced images in practice." [Carr and Hartley, 2009]. As an example for Tan's method in terms of image quality, please refer to figure 3.2.

3.6. The Deep Photo System

A rather different approach was proposed by Kopf et al. in 2008 called the *Deep Photo System*. Crucial to all dehazing methods is to acquire the depth information of every pixel in the frame, but rather than acquiring these by making assumptions or employing statistical observations, Kopf et al. developed a data-driven dehazing procedure, by employing a registration process to align the photograph within an existing 3D model. This way the method does not need to estimate the distances in the scene, but will get the exact distances right away, assuming that such kind of georeferenced digital terrain and urban models are available. They propose a user interactive referencing system, in which the user registers certain scene points with the corresponding points in the model, such a model could come from satellite image data obtained from GoogleEarth™,

BingMaps^{TM}or other providers. The 3D models for buildings and other objects are already available for many cities like Berlin or New York City and others(for example through VirtualEarth^{TM}). Additionally helpful can be GPS tags, produced by the imaging system itself, sometimes even tilt and heading is provided by these cameras. With a static surveillance camera in mind, the scene must be georeferenced only once initially and could then rely on a set of depth information indefinitely assuming no camera shifts take place. Taking airports as an example, 3D models of buildings and high resolution geo information is usually available due to construction plans and air traffic controlling agencies. Here also lies the limitation of such a system, since it heavily relies on those sets of data, if no 3D model of the scene is available, no dehazing can be performed. Also in dynamic scenes, with vehicles such as aircrafts driving or flying through, there is no depth information for those foreground objects whatsoever, hence dehazing for those objects would rely on wrong depth information and cause a superabundant dehazing on those spots. However, for scenes where distances are great and the influence of moving objects to scene distance is low, such as in figure 3.2, the image quality is remarkable. After acquiring the depth information, the method of Kopf et al. estimates the airlight and the attenuation coefficient similarly to the other haze removal methods and then basically solves Koschmieder's equation, however with some nuances differently. In the paper [Kopf et al., 2008], the authors explore further possible applications for the depth information of an image, other than dehazing, such as approximating changes in lighting, expanding field of view, adding haze, adding new objects into the image with the correct haze values according to distance, and integration of GIS data into the photo browser, just to name a few. Since often the depth map of other dehazing methods also comes as a byproduct, these mentioned applications may also be implemented combined with other dehazers, such as [Fattal, 2008] or [He et al., 2010a], for example. Problems may arise from the fact that the alignment between the photograph and the model may not be completely accurate due to unprecise 3D models or the lens curvature of the imaging system.

3.7. Improved Dark Object Subtraction

The *Dark Object Subtraction* is a method widely used in multi-spectral remote sensing systems and an improved version of it was inspiring for the dehazing method in the

next section, the *Dark Channel Prior*, which is why DOS(Dark Object Subtraction) will now be summarised briefly. The DOS was developed many years before the improved version, but is now almost entirely used in its improved version today, as it was introduced by Chavez in 1988([Chavez, 1988]). The main idea behind this method is that in a scene at least one dark object exists, that has zero reflectance(is black). Also any measured radiance is attributed to atmospheric path radiance only. This also means, that the atmospheric transmittance is not corrected. The method needs at first to identify a dark object in the scene and then estimate the atmospheric light, which is the additive term to the otherwise black pixel(if input image were haze free). This constant is then subtracted from all pixels in the image in order to remove the first-order scattering component. However in satellite imaging, where this technique is mostly used, the imaging sensor usually provides multi spectral data, such as from the Landsat Thematic Mapper, thus in the improved version the spectral bands are no longer treated separately from the scattering properties but dependent on the specific wavelength-scattering relationship(see Mie Scattering, page 42), since atmospheric scattering is highly wavelength-dependent. This leads to much better results when using colour images, since each colour band will have its own dehazing iteration. Chavez's method "allows the user to select a relative atmospheric scattering model to predict the haze values for all the spectral bands from a selected starting band haze value. The improved method normalizes the predicted haze values for the different gain and offset parameters used by the imaging system." [Chavez, 1988]. This was also one of the first dehazing methods that does not need further information about the scene than those already contained within the image data. Difficulties arise in scenes where no dark object is present, which is a rare case in satellite images with large amounts of pixels, where it is statistically very likely to have also dark pixels, arising from shadowed areas. However these scenarios where dark objects are missing do exist. The performance of this method can be seen in figure 3.1.

3.8. Dark Channel Prior

The success of recently developed techniques such as [Fattal, 2008], [Kopf et al., 2008] and [Tan, 2008] compared to earlier dehazing methods lies in using stronger assumptions. A very promising new single image technology, developed in 2010 called the *Dark Channel Prior* comes from He, Sun and Tang. This method does not rely on significant

variance on transmission or surface shading in the input image and the output image is less effected by halos than in [Tan, 2008]. Although every assumption limits the algorithm to specific use cases, the main assumption here seems to work for most outdoor scenes, except for those where "the scene object is inherently similar to the airlight over a large location and no shadow is cast on the object" [He et al., 2010a]. The main prior in this method is, as the name lets assume, the dark channel prior, which is a statistical based assumption of haze-free outdoor images. The prior says, that in most of the local regions that aren't sky, very often some pixels have a very low intensity in at least one of its colour channels(RGB). In the hazy image then, these dark pixels can be used to determine the true airlight, since the airlight is apparent on a dark object(as stated in the preceding chapter). The dark channel J^{dark} of J(the haze-free image) is defined as

$$J^{dark}(x) = \min_{c \in \{r,g,b\}} \left(\min_{y \in \Omega(x)} (J^c(y)) \right) \tag{3.15}$$

where J^c is a colour channel of J and $\Omega(x)$ is a local patch centered at x. This statistical observation is called the *dark channel prior*. These low intensities come from natural phenomena such as shadows or just really dark or colourful surfaces. Since J^{dark} tends to be zero and as A^c, the corresponding channel of the atmospheric light is always positive, it may be written:

$$J^{dark}(x) = \min_{c \in \{r,g,b\}} \left(\min_{y \in \Omega(x)} \frac{J^c(y)}{A^c} \right) = 0. \tag{3.16}$$

This can be used to estimate the transmission for that patch $\Omega(x)$ by putting 3.16 into the image formation model 3.12, however now in combination with the min operator:

$$\min_c \left(\min_{y \in \Omega(x)} I^c(y) \right) = \widetilde{t}(x) \min_c \left(\min_{y \in \Omega(x)} J^c(y) \right) + (1 - \widetilde{t}(x)) \cdot A^c. \tag{3.17}$$

with $\widetilde{t}(x)$ denoting the transmission in a local patch, then putting 3.16 into 3.17 leads to:

$$\widetilde{t}(x) = 1 - \min_{c} \left(\min_{y \in \Omega(x)} \frac{I^c(y)}{A^c} \right) \tag{3.18}$$

which is a direct estimation of the transmission for each local patch. They then apply a soft matting algorithm on the depth map, this leads to a much smoother and detailed depth map, an example of that can be seen in figure 3.3(b). Having the transmission or depth map, the scene radiance according to 3.12 can now be recovered. However, since the direct attenuation term $J(x)t(x)$ can be very close to zero, the transmission is restricted to a lower bound t_0 for example $t_0 = 0.1$, since the scene radiance is typically not as bright as the atmospheric light A. The final scene radiance $J(x)$ may then be recovered by:

$$J(x) = \frac{I(x) - A}{\max(t(x), t_0)} + A. \tag{3.19}$$

In the above calculations, the atmospheric light A was considered to be known, which is of course not the case, at least initially. Unlike other workers in the field, He et al. do not take the pixel with the highest intensity as the atmospheric light, since this could as well be a white surface such as a white airplane or a bright building veneer. He et al. pick the top 0.1% brightest pixels in the dark channel($\min_c(I^c)$), since these must be the most haze-opaque. Amoung these pixels, the pixel with the highest intensity in the input I is picked as the atmospheric light A. This may not be the brightest pixel in the image, but is more robust than the "brightest pixel" method according to [He et al., 2010a]. This method seems very elegant and shows very good results, as one can see from figure 3.3.

3.9. Geometry Based Dehazing Methods

This technique described by [Carr and Hartley, 2009] is more an improvement that can be used throughout all single image based dehazing methods than a dedicated image dehazing method itself. Thus this principle can be used in combination with the methods mentioned above. The authors Carr and Hartley investigated existing dehazing methods and introduced the idea of simplifying the depths map estimation by assuming a common geometry that applies for most outdoor surveillance scenarios. Their motivation

can be best described by a citation: "Although each method[meaning existing dehazing methods] uses a different statistical measure to drive the estimation process[depths estimation], they all share a common shortcoming: when the appearance information of a pixel is unreliable, the algorithms are unable to produce a good depth estimate for the corresponding location in the image. " [Carr and Hartley, 2009]. The first assumption is, that neighbouring pixels have similar depths, which really isn't new to dehazing methods, but is used in most methods. However, the new assumption is, that for outdoor surveillance cameras, a geometry like in figure 3.9(a) can be assumed. Or in words, the geometry of a camera located above the ground, high in the air and tilted towards the ground. The resulting simple relationship is, that objects which appear closer to the top of the image are usually further away, see figure 3.9(b).

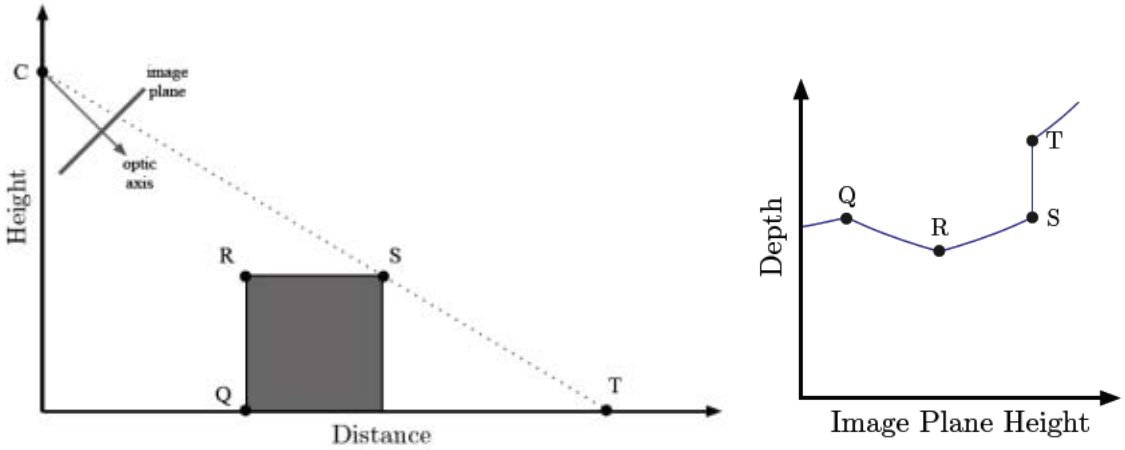


Figure 3.9.: Assumed camera geometry, the depth of any scene point can be split into distance and height components. If the scene does not contain any cave-like surfaces, the distance of scene points will increase monotonically from the bottom of the image to the top, *Source:* [Carr and Hartley, 2009]

As the described dehazing methods are using regularisation and optimisation methods, so does this method. Here the graph-cut based α-expansion method is used, as employed by [Tan, 2008]. The assumptions of the geometry are herein used as a preference rather than a hard constraint and can be used within an energy minimisation framework such as used by [He et al., 2010a]. With this as only a soft constraint it even allows for the, in outdoor surveillance unusual, cases in which objects in the top of the image are closer than they should be according to the assumptions. With a static camera position, the depth map can be calculated once and used for several frames in which the atmo-

spheric properties remain roughly the same. A motivation is given by the authors of this method: "The transportation industry would benefit from automatic fog enhancement technology. For this application, the primary interest is enhancing video - not single images." [Carr and Hartley, 2009]. They also state, that they could perform real-time image dehazing using gpu's(graphics processing units). For this thesis, these are important statements, since a simplification was introduced for static cameras and also a validation that real-time dehazing is in fact possible, however using the GPU rather than a CPU. It can be learned even more from the experiences of Carr and Hartley: "If the camera is positioned high in the air, the difference in depth between a foreground object and the background behind it is usually small. Therefore, one can typically estimate the depth of the background image and apply this to any video frame without creating significant errors in the enhanced image.", this principle will be used in the successive section for the development of the real-time dehazer. However, the error being small, may be greater when estimating the depth map with too much foreground in the scene, such as moving objects occluding parts of otherwise distant surfaces. This would cause an under estimate for those spots in the scene when the background becomes visible again. Due to moving foreground objects it may be convenient to renew the depth map every certain number of frames or when alterations in the scene have been detected by a movement detection algorithm for example. The consequences of using the monotonic depth assumptions compared to an unconstrained method is illustrated in figure 3.10, here the authors Carr and Hartley used a surveillance camera video as input and used the dark channel prior method to enhance the frames in combination with their depth map optimisation method.

As a side note, Carr and Hartley used a gamma correction before applying their dehazing algorithm to the image, as described in an earlier section. Which improved the overall contrast, making the work of the dehazer more effective.

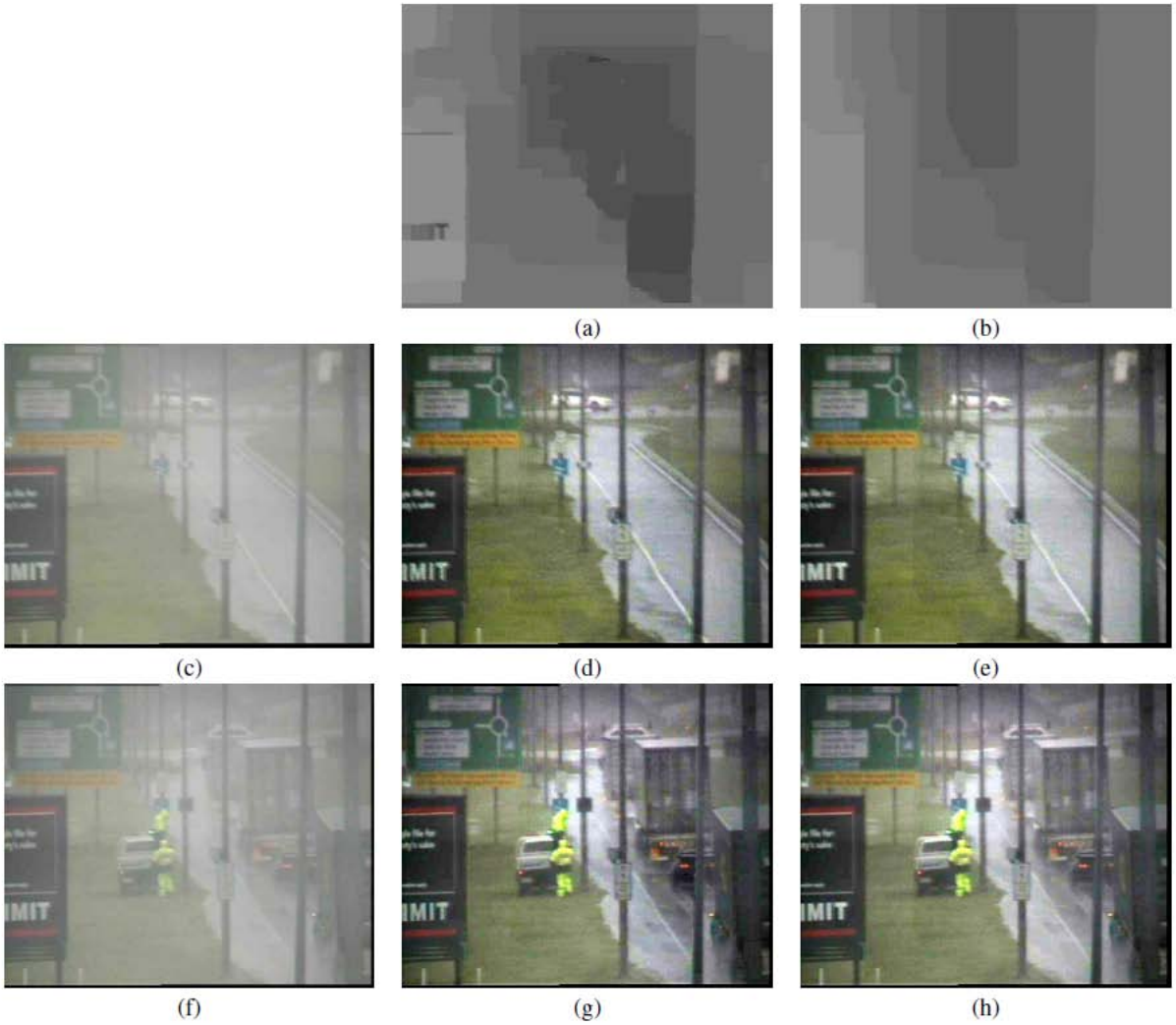


Figure 3.10.: Video frames (c) and (f) are enhanced using depth maps (a) and (b) which were estimated without and with monotonicity respectively. Both depth maps were estimated using the contrast data model applied to video frame (c), as the dark channel prior has difficulty with roads. The unconstrained depth map (a) has outliers (due to the appearance of foreground objects and an inherent difficulty with the textureless road), but these do not induce significant artifacts in the enhancement (d). Since the artifacts are not significant, the improvement in the enhancement (e) from using a monotonic depth map (b) is not substantial. However, when the depth map is applied to a video frame (f) captured later, the outliers in (a) over-enhanced the middle vehicle beyond the dynamic range of the image (g). This does not happen in the monotonic correction (h). Although these artifacts may be subtle in a still image, they are quite apparent in a video. *Source:* [Carr and Hartley, 2009]

4. Real-Time Image Dehazing

This chapter describes the dehazing program developed for this thesis. Quoting Namer: "This work is thus part of a process of making the method more reliable and useful" [Namer and Schechner, 2005], talking about the image dehazing and polarisation-based dehazing in particular. So is this work, dehazing methods came a long way from Koschmieder's theory and the development of fast computer technology as well as electronic video sensors. Real-time image dehazing is rather new to this subject and thus still in the process of development. For this thesis a dehazing algorithm was implemented with new improvements that make it possible to dehaze an image in real-time utilising just the CPU. In this chapter, the choice of the algorithm basis is explained. In the next step, the working principles of this implementation with a description of the improvements is given. Followed by a section discussing several brightness enhancement methods, and a section presenting the parameters of the algorithm. The chapter is rounded off by a section about possible further directions of research.

4.1. Choosing the Algorithm

In the preceding chapter many dehazing algorithms have been described, although most presented algorithms give excellent visibility improvements, one problem remains. The calculation time per image is too long for real-time applications. The author chose the algorithm that had the biggest potential for speed improvements. Chosen was the Dark-Channel Prior method, which allows a high degree of parallelisation, is straight forward, logical, works with most scenarios, not just special cases like only certain scene geometries and also works on grey scale images, which is relevant to real world applications. It is also important that the required user interaction is minimal, unlike for example [Narasimhan and Nayar, 2003]. The results of He et al.'s method were already shown to

be at least as good as other methods in direct comparison with other dehazing methods in terms of image quality and degree of dehazing or defogging, respectively. Other researchers also chose He et al.'s algorithm as basis for their implementation for a real-time image dehazing program. In the paper [Lv et al., 2010], Lv et al. describes the implementation of He's method utilising the graphics processing unit of a computer. A comparison to Lv et al.'s implementation will be given in chapter 5, page 99.

4.2. Dark-Channel Prior Implementation

The implementation follows in general the flow process of He et al.'s algorithm, which can be seen in figure 4.1.

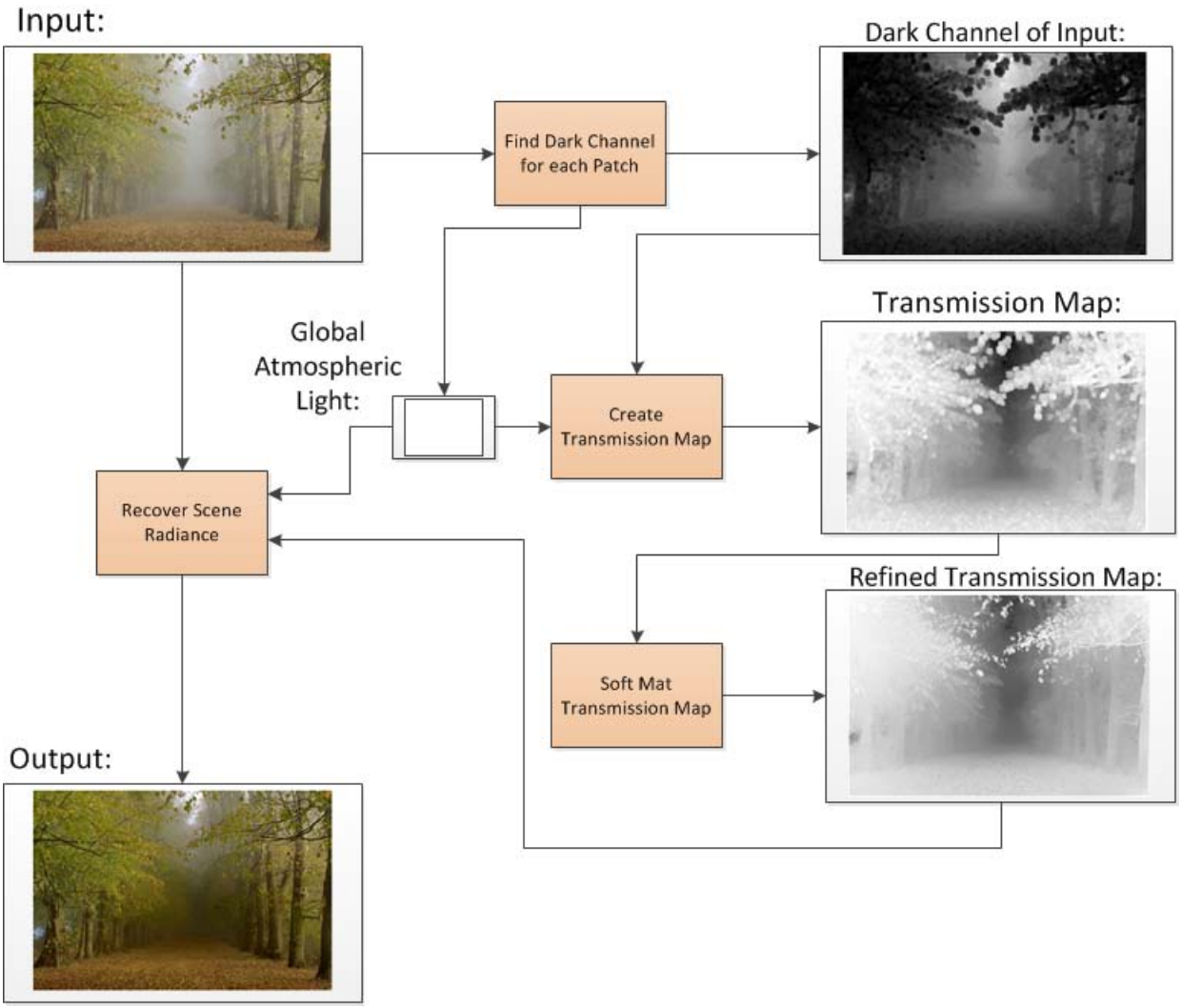


Figure 4.1.: Process flow chart of He et al.'s dehazing method. The patchsize in this example is set to 15x15. The example image is taken from [He et al., 2010a]

A component wise analysis of He et al.'s method shows that the processing time predominantly depends on the soft matting algorithm, therefore reducing or even eliminating the processing time of the soft processing would improve the overall dehazing time greatly. More than any attempt to reduce calculation time in one of the other components. The main idea behind the speed improvement is to eliminate the soft matting step of the transmission map, it is based on the assumption that: "Typically, errors in a depth map do not induce significant artifacts into the enhanced image [..], since the solution is applied to the data used during estimation." [Carr and Hartley, 2009]. In order to achieve satisfying results, the patchsize must be reduced to about 4x4 pixels instead of the pro-

posed 15x15 pixels by He et al. This way blocky artifacts are reduced when leaving out the soft matting step. Generating an unfiltered transmission map is of magnitudes faster than generating a filtered(e.g. soft matted) transmission map. The other steps of the dehazing process can be kept roughly the same. The following chart(4.2) shows the process flow of the authors dehazer.

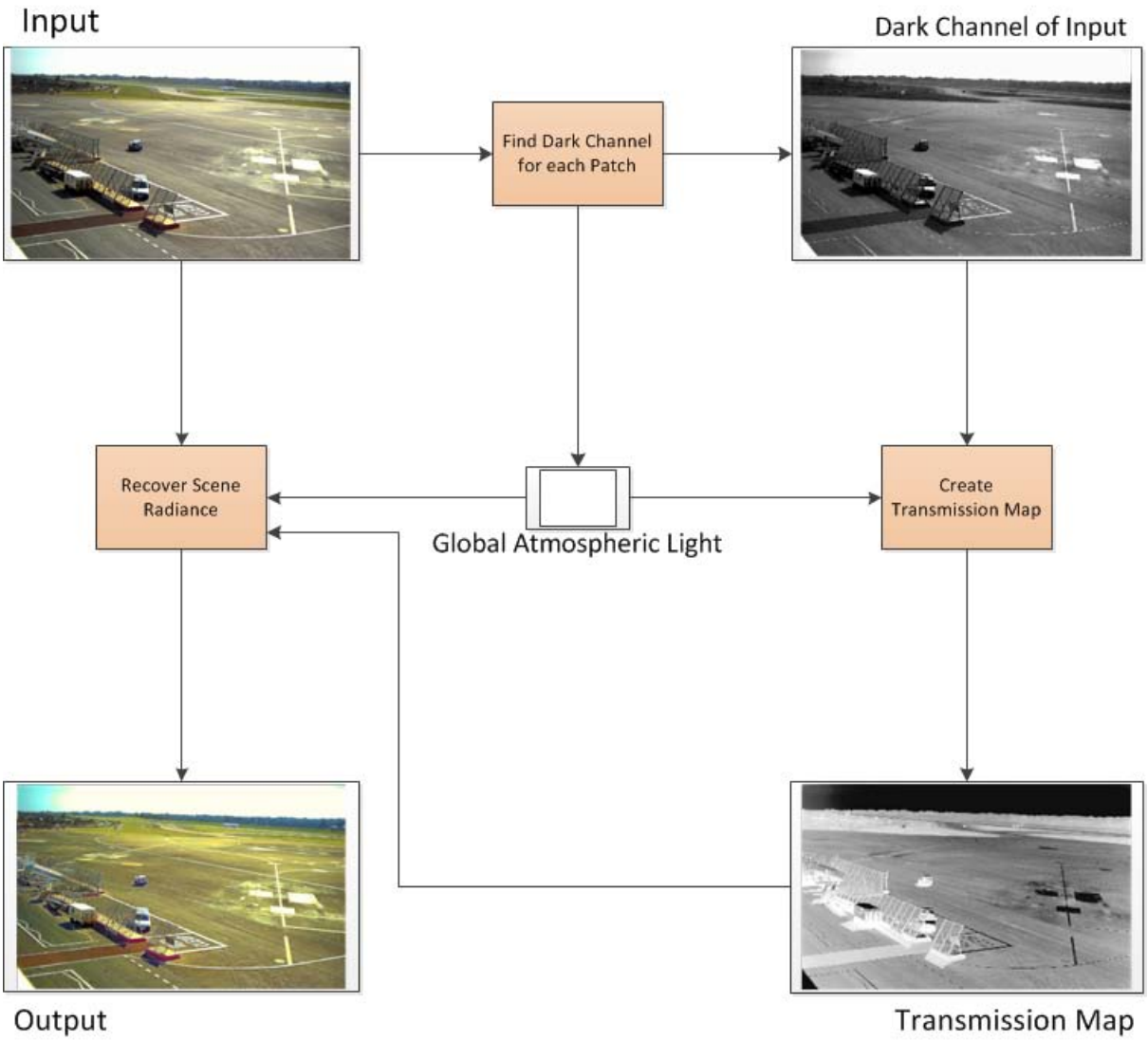


Figure 4.2.: Process flow chart of the dehazing method. The patchsize in this example is set to 4x4. The example image was taken at Malta International Airport under clear weather conditions.

Three main modules can be seen in the above flow chart, all are called from the function "Dehaze()"(appendix A, page 117). The first module to enable during the dehazing

process is the "Find Dark Channel for each Patch" module. It is the realisation of equation 3.16(page 68):

$$J^{dark}(x) = \min_{c \in \{r,g,b\}} \left(\min_{y \in \Omega(x)} (J^c(y)) \right). \tag{4.1}$$

The goal is to find the dark channel of each pixel, then finding the darkest one within the specified patch. While this step is performed, the pixel that created the dark channel for the patch is stored in a queue. This queue will be used to determine the atmospheric light of the image as soon as all dark channels have been found. The implementation of the dark channel finding module can be seen in the appendix A on page 119. The function for the single thread version is the most readable, however the faster multi threading version can also be found in that section of the appendix. After all dark channels have been found, each patch will be given its monochrome colour. Now the atmospheric light can be calculated, this is done by the function "CalcAtmosphericLight()"(appendix A, page 131). There are actually many ways to determine the atmospheric light of an image. In Tan's work [Tan, 2008], the value with the highest intensity is chosen, Fattal computes the airlight by solving an optimisation problem [Fattal, 2008], others just set the airlight to (1,1,1) after performing a white balance to the hazy image, however the author chose to implement He et al's robust version by determining the atmospheric light as the brightest pixel among all pixels in the dark channel. The data structure for the queue can be found in the appendix B on page 138 and in appendix C on page 145.

The second module in flow chart 4.2 is the "Create Transmission Map" module, it is called after the dark channel and the atmospheric light have been retrieved. This module basically is the realisation of the equation:

$$\widetilde{t}(x) = 1 - \min_{c} \left(\min_{y \in \Omega(x)} \frac{I^c(y)}{A^c} \right), \tag{4.2}$$

which is the same as 3.18 from page 69. This calculation is performed by the function "CreateTransmissionMap(..)", to find in appendix A, page 127. The third and last module is the "Recover Scene Radiance" module, it puts all 3 parts, namely the transmission map, the input and the atmospheric light together and implements the equation:

$$J(x) = \frac{I(x) - A}{\max(t(x), t_0)} + A, \tag{4.3}$$

which is the same as equation 3.19(page 69). This step is done by the function "Create-Output(..)", to find in appendix A, page 132, also in that section of the appendix is the multi threading version of that function. All other sub routines needed for the dehazing process can also be found in the appendix.

An implementation based on the process flow 4.2 is already 1-2 orders of magnitude faster than He et al.'s original implementation. However, further speed improvements can be done by parallelisation in the form of multi threading. All modules can be divided into threads, one for each core of the CPU, each operating on a subset of the original image. Now, since the transmission map creation is no longer dependent on neighbouring pixels, this step can be performed parallelised as well. In this implementation the images are divided horizontally in equally large subsets. Flowchart 4.3 shows the working principle for a CPU with 4 cores as example, but it could as well be any other number of cores.

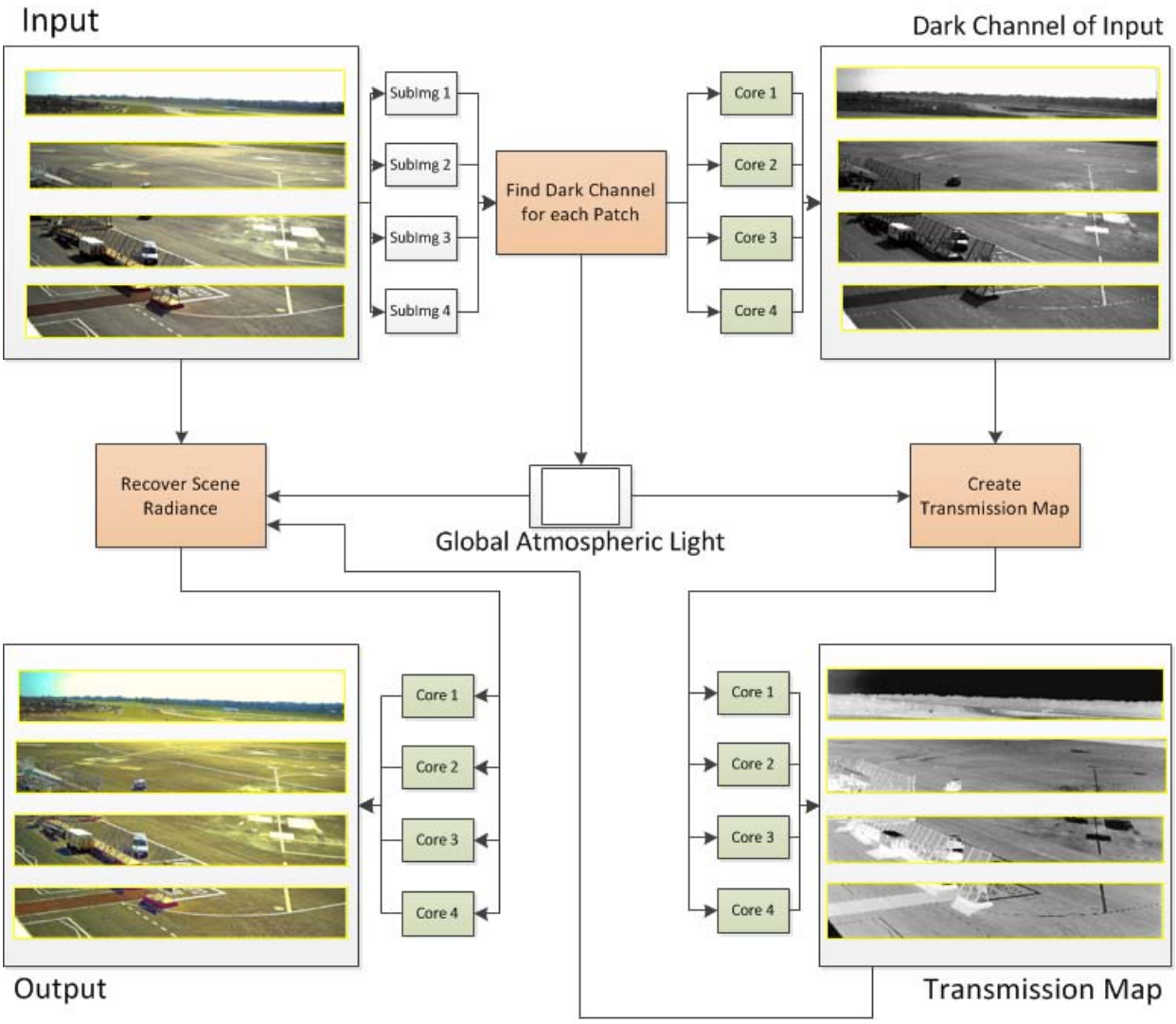
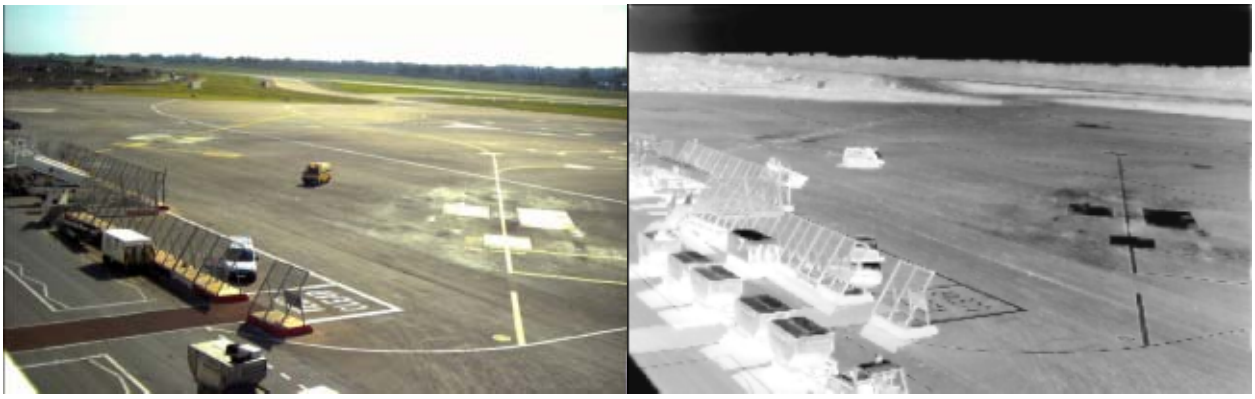


Figure 4.3.: Process flow chart of the multi-threaded dehazing method. The patchsize in this example is set to 4x4. The example image was taken at Malta International Airport under clear weather conditions.

A second idea is to calculate a filtered transmission map only once every x frames, where x could be any integer greater than 1. This way, the costly calculation of a soft matted or otherwise refined transmission map could be done in the background without compromising the dehazing of the current work. A couple of filters were tested for this matter. First, the soft matting algorithm used by He et al. and proposed by Levin [Levin et al., 2006], which gives the best results from all tested filter techniques but is also the second slowest with about 5 seconds on the test machine(the test machine is described in detail in the following chapter) for the transmission map of a 1376x856 large image. Second, the

Gauss-Markov random field model guided by the input colour of the input image as used by Fattal is the slowest tested here, with about 6 seconds on an 1376x856 large image, but gives almost as exquisite results as Levin's filter. However, updating the transmission map only every 5-6 seconds introduces significant errors when the scene contains moving parts, such as airplanes or cars. This phenomena was already addressed by Carr, who said: "Here, the depth map is estimated using one video frame, but applied to another. As a result, the estimation error caused by various foreground appearances become quite apparent in the enhanced images[..]." [Carr and Hartley, 2009]. The effects of that can be seen in figure 4.4.



(a) Input (b) Transmission Map

(c) Output

Figure 4.4.: Figure demonstrates the effects of a delayed transmission map. 4.4a shows the current input image and 4.4c shows the dehazed output that assumed the transmission map given in 4.4b. The example image was taken at Malta International Airport.

As can be seen from figure 4.4, those spots that are supposed to be foreground, but are still background due to the old transmission map, are overestimated in terms of distance during the dehazing, whereas spots where foreground has become background get underestimated. This results in hazy and too bright spots where for example a vehicle stood when the transmission map was created but has left now. On the other

hand, wrong or pale colours are created where a vehicle is now located, which was not present when the transmission map got created. This is a confirmation of Carr's finding. The consequence is that the transmission map must be up to date for the dehazing step. One could think of an adaptive way of transmission calculation, where only the parts of the transmission map will be updated where a new object was detected or where pixels have changed significantly from one frame to the next, since changing pixels means that the depths at that points must have been changed and therefore the transmission map is now invalid for at least those points. However, tests have shown that in scenes where large parts of the input image changes, due to large aircrafts or many small vehicles, the processing time increases significantly and converges towards the full transmission map processing time of 5-6 seconds per frame depending on the implemented filtering. Also due to changes in brightness of the camera and noise, respectively, from frame to frame, significant artifacts can be introduced, such as those shown in figure 4.5.

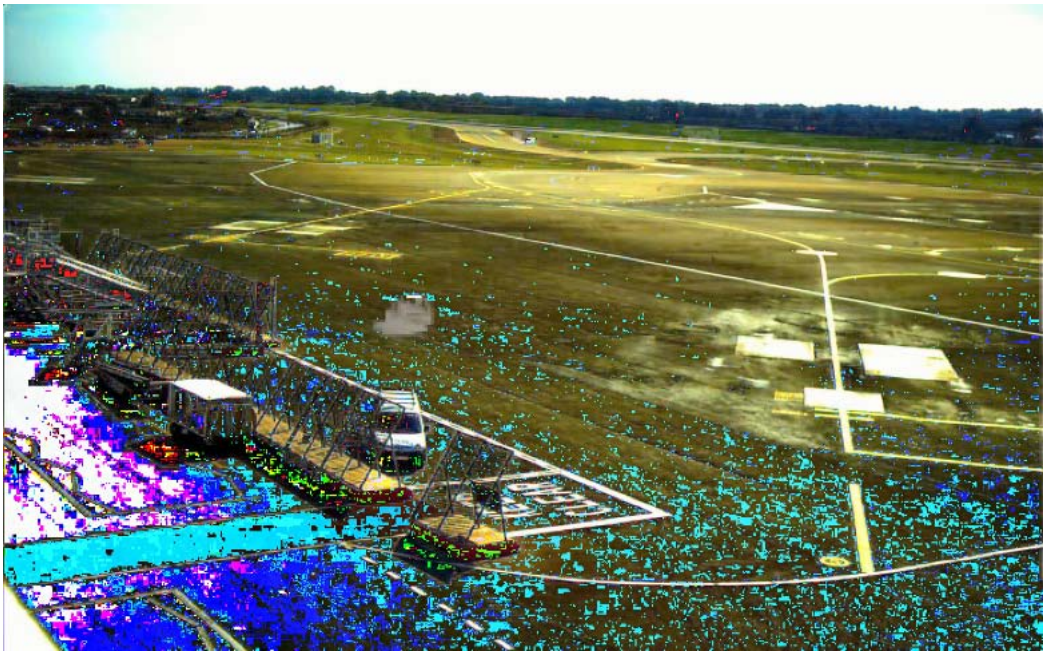
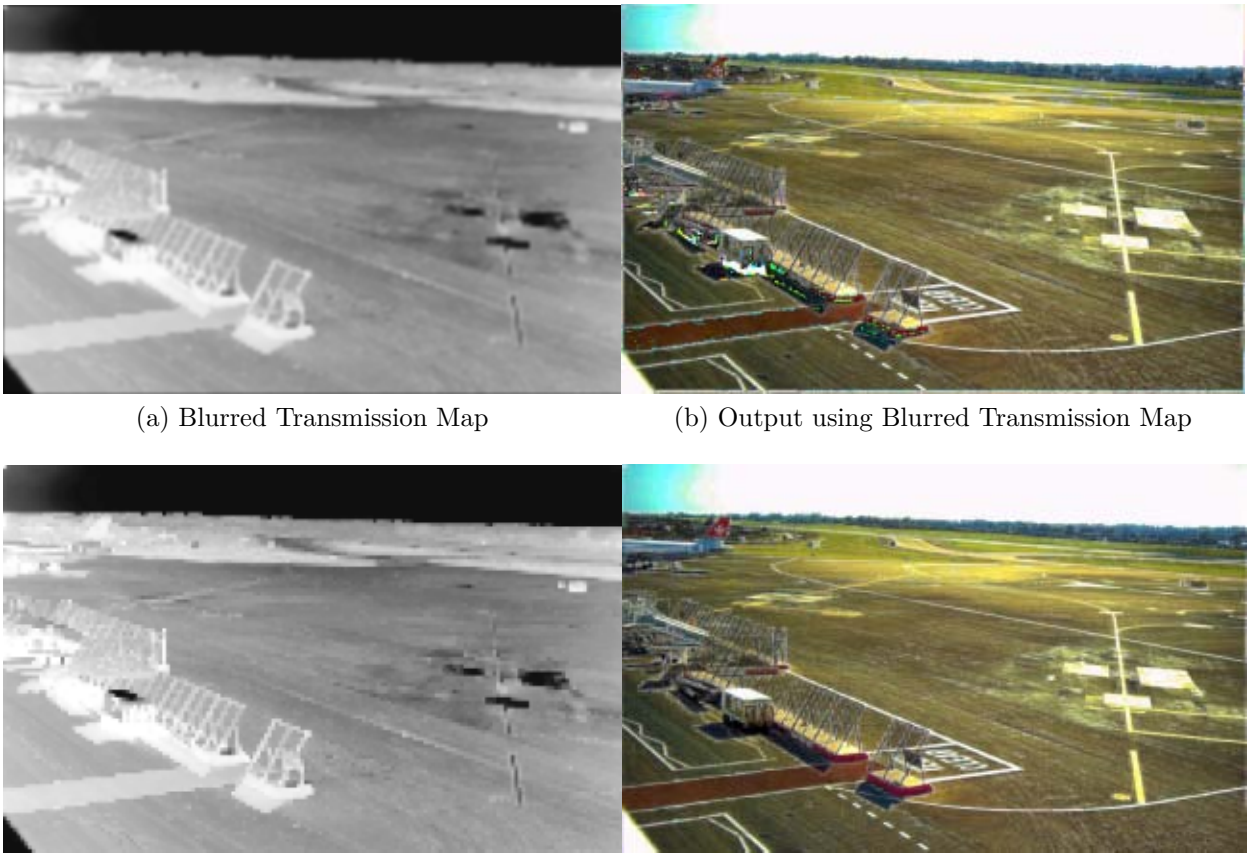


Figure 4.5.: Figure shows the possible artifacts in an output image of the dehazer when an old transmission map is used.

In the above example in figure 4.5, a cloud casting a shadow on the ground is responsible for huge amounts of artifacts after just 5 frames. Due to these findings, it was implemented a Gauss filter for the transmission map, in a way that the overall dehazing time does not exceed 1 second on the test machine. 1Hz is the least acceptable frame

rate for a real-time application such as air traffic surveillance systems. The goal of the filtering is to reduce the block effects of an unfiltered transmission map. However, since the common Gauss filter does not preserve sharp edges, but in fact smoothens them, the block effects are only traded in for halos, as shown in figure 4.6.



(a) Blurred Transmission Map

(b) Output using Blurred Transmission Map

(c) Unfiltered Transmission Map

(d) Output using Unfiltered Transmission Map

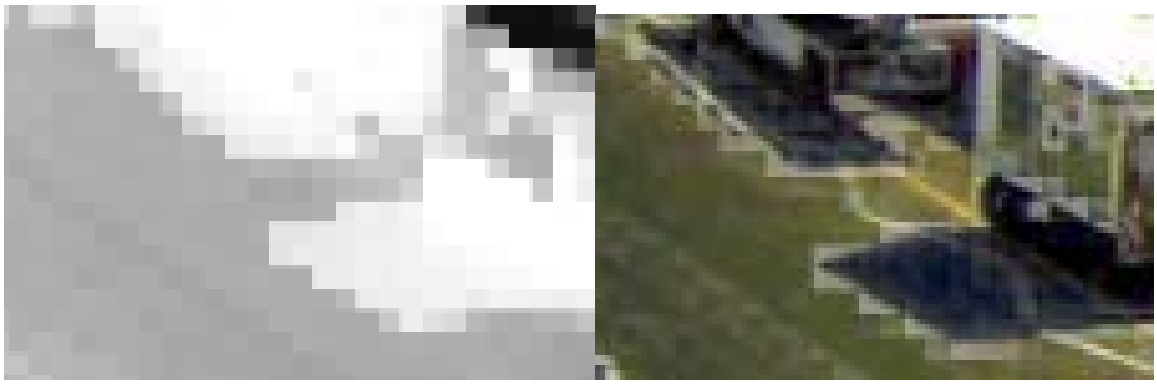
Figure 4.6.: Figure shows the use of a filtered transmission map, in this case the Gauss blur with radius 5 and a patchsize of 8x8. The top row contains the blurred transmission map and the resulting output, for comparison, the bottom row contains the unfiltered transmission map with the same patchsize of the same input and its resulting output.

Although the output image now lost the block effects, the halos are more apparent, even though smooth. However, visible in the output image are also artifacts in form of single over saturated colour defects due to the blurred transmission map. Several sets of parameters were tested, no combination of gauss kernels and radius satisfied the requirements. Alterations in the parameters always show the trade-off between halo

effects and block effects versus colour defects. It is crucial that edges remain sharp in the transmission map in order to avoid halo and colour defects. Also, it is desirable to have large patchsizes in the transmission map, since only this way the dark channel assumption may be satisfied. Therefore another approach was implemented where 2 transmission maps, each with a different patchsize were blended. Several combinations were tested, the best results were performed by 2 transmission maps, one with patchsize 1 blended with another with patchsize 8. This filter is also a low pass filter that reduces noise and reduces the block effects of the transmission map with larger patchsize. The results can be seen in the next figure 4.7.

(a) Blurred Transmission Map (b) Output using Blurred Transmission Map

(c) Unfiltered Transmission Map 1x1 (d) Output using Unfiltered Transmission Map 1x1



(e) Unfiltered Transmission Map 8x8 (f) Output using Unfiltered Transmission Map 8x8

Figure 4.7.: Figure shows the use of two blurred transmission map, the first with patch-size 1 and the second with patchsize 8. The top row contains the blurred transmission map and the resulting output, for comparison, the middle row contains the unfiltered transmission map with patchsize 1 of the same input and its resulting output. The bottom row contains the transmission map and dehazed image for the patchsize of 8.

The transmission maps are weighted, the map with patchsize 8 is only weighted 30%, whereas the map with patchsize 1 is weighted 70%, this way block effects are avoided. Artifacts are not introduced by this method unlike the Gauss blur. With other weights, block effects may still be apparent. As one can see, best contrast enhancement can be optained from transmission maps with large patchsizes, as an example see the zigzag line in the shadow of the van in figure 4.7, which is best visible in the unfiltered transmission map with 8x8 patches. This effect is due to the dark channel prior. This filter is however only optional in the final program, since the improvement is not always necessary and takes up processing time. Also, non-filtering already gives satisfying results.

4.3. Brightness Enhancement

The output of this algorithm looks dim, hence several approaches were taken to raise the level of brightness. The first approach is a histogram equalisation, as described in the preceding chapter on page 56, but since most of the examined scenes have both some pixels with close low luminance values and pixels with close high luminance values but both in small quantities, standard histogram equalisation can only improve contrast in either one, high or low luminance areas. Histogram Equalisation takes into account the amount of pixels that uses a specific brightness value, if like in figure 4.7, only very small patches are using specific brightness values, then those patches may not profit from the contrast enhancement, although the overall contrast of the image may be improved. The second approach is a histogram splay, this method improves contrasts by stretching the used luminance values over the entire spectrum of possible luminance values, however does not takes frequencies of specific values into account. This however only improves contrast in scenes where the full spectrum is not already used. Best results both in terms of visibility enhancement and processing speed was done by the simple gamma correction, as described too in the preceding chapter on page 56. A comparison between these 3 methods can be seen in figure 4.8.

(a) Input Image

(b) Direct Output

(c) Histogram Equalisation

(d) Histogram Splay

(e) Gamma Correction

Figure 4.8.: Figure shows the use of different brightness correction methods. 4.8a shows the hazy input image and 4.8b shows the dim looking output without any brightness correction. The last 3 images show 3 different methods applied to the direct output, 4.8c illustrates the histogram equalisation, 4.8d illustrates the histogram splay and finally 4.8e shows the simple gamma correction. The example image was taken at Abu Dhabi airport.

Apparent from the images is that histogram equalisation and histogram splay amplify the effects of the blocks from the jpeg compression. Because the dehazing improves contrast of the image, invisible patches with apparently the same colour in the input can get very apparent colour differences in the output, which accentuate the jpeg patches with same colours. The contrast enhancer can be applied to both the input image, before calculating the dark channel and to the output image before displaying it to the user. However applying it to the input image amplifies the block effects originating from the unrefined transmission map, as shown in figure 4.9.

Figure 4.9.: Figure shows the dehazed image, used was the histogram splay method performed before dehazing on the input image.

It has been found by accident by the author that a third option is to apply the histogram function to the transmission map, which improves the contrast in the transmission map, and this in the dehazed output. There is no theoretical basis however why this should improve image quality. The effect is demonstrated in figure 4.10.

(a) Input Image

(b) Dehazed Output

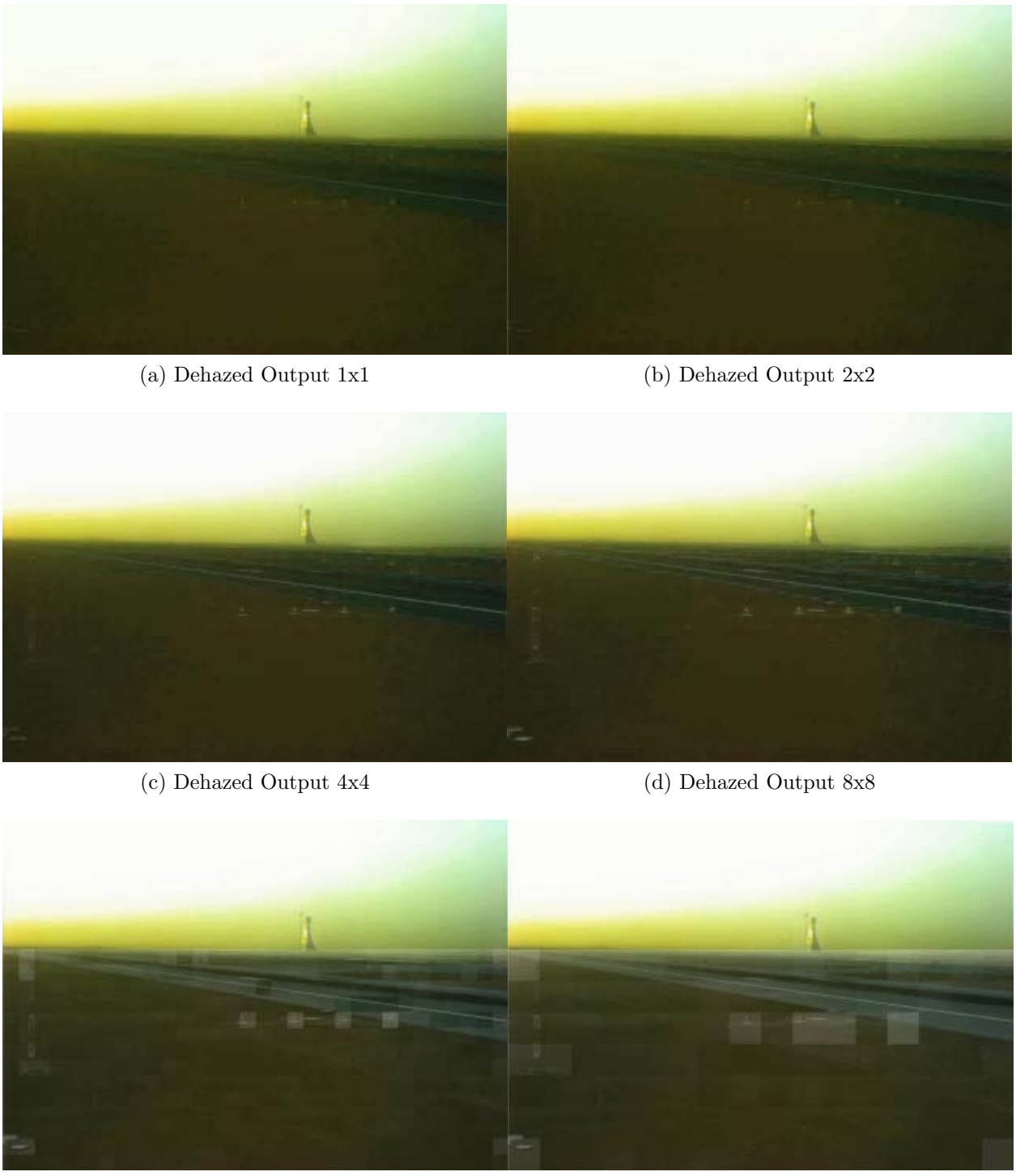
(c) Dehazed Output with Equalised Transmission Map

Figure 4.10.: Figure shows the hazy input image and the dehazed output images. Figure 4.10b shows the dehazed image with the common dehazing method described above, and figure 4.10c shows the new approach where a histogram equalisation was applied to the transmission map. The example is a segment of an image that was taken at Zurich airport, Austria in bad weather conditions(heavy snow storm). Note: the input image is compromised by snowflakes on the lens, for example in the top middle and bottom right.

Although distant objects become more distinguishable from their surroundings(e.g. the plane in the above figure at about three quarter of the image to the left), the colours become more over saturated and unnatural. Also visible in 4.10c, some colour artifacts are introduced due to the altered transmission map. As a consequence, the transmission map now produces overestimated and underestimated transmissions, respectively. The contrast enhancement methods mentioned in this section can all be found in the appendix D.

4.4. Program Parameters

Several parameters can be tweaked in order to achieve satisfying dehazing results. First, the patchsize of the dark channel and transmission map is variable. The larger the patchsize is, the more likely it is that the dark channel prior is satisfied, hence this will assure stronger dehazing. However, larger patches means that the transmission map may be wrong, since the transmission will not always be constant within a patch. The smaller the patchsize is, the smaller are the visible errors in the dehazed image, especially those along edges. Figure 4.11 shows the effects of different patchsizes.



(a) Dehazed Output 1x1

(b) Dehazed Output 2x2

(c) Dehazed Output 4x4

(d) Dehazed Output 8x8

(e) Dehazed Output 43x43

(f) Dehazed Output 172x172

Figure 4.11.: Figure shows the effects of increased patchsize to the output image. The example image has the dimensions 1376x856 and was taken at sunset under hazy weather conditions at Abu Dhabi airport.

Step-effects along diagonal lines, such as the runway center line, become more visible with small patchsizes. The larger a patch becomes, the more obvious the colour discontinuities from one patch to another. Also to see in the figures are bright patches around objects with a very low dark channel. The image gets brighter with increasing patchsize due to the dark channel prior. The right choice of patchsize depends on the input scene. Possible patchsizes naturally depend on the dimensions of the input image.

A second parameter stems from equation 4.3, page 78. The parameter t_0 defines the lower bound of transmission. When choosing the parameter to be over 100% the application is capable of adding haze to the scene. With t_0 being 0% it has been found in some cases, that artifacts are introduced to the dehazed image due to noise. Also jpeg compression limitations combined with $t0 = 0$ cause blocks around sharp edges to become more visible. Also with $t_0 = 0\%$, the sky or other very bright regions are forced to take on a slightly darker colour. Henceforth, choosing a t_0 above zero is also a noise cancelling action. The effects of an altered lower transmission bound can be seen in figure 4.12.

(a) Input - snippet

(b) Input

(c) Output snippet

(d) Output, lower bound = 0%

(e) Output snippet

(f) Output, lower bound = 15%

(g) Output snippet

(h) Output, lower bound = 50%

Figure 4.12.: Figure shows the effects of increased lower transmission bound. The example image was taken at Cologne airport under light wet fog condition.

Although all transmission values below 50% are set to 50% in figure 4.12h, colours of objects close to the camera still appear more vivid than in the input, but the disadvantage of lowering the lower transmission bound is that the dehazing is less effective in the upper parts of the image where the distance is greater. In most scenes a lower bound of about 10%-15% transmission has been found to be a reasonable value.

The third parameter regulates the degree of dehazing, He et al. called that parameter ω, which is a real number between zero and one. It assures that more haze is kept at more distant objects. The parameter can be applied to equation 3.18(page 69), which gives:

$$\widetilde{t}(x) = 1 - \omega \cdot \min_{c} \left(\min_{y \in \Omega(x)} \frac{I^c(y)}{A^c} \right). \tag{4.4}$$

The introduction of this parameter has two reasons. Firstly, completely haze free images may seem unnatural and the feeling of depth may be lost for the viewer due to the effect called *aerial perspective*. Secondly, the physical background of the dark channel prior actually assumes an haze free image, however manages to handle certain degrees of noise well. In some rare cases during testing an ω of 1.0(100% dehazing) introduced artifacts and heavily amplified noise effects from the input image. Since the input image is never free of haze, ω has been found to give reasonable results at about 95% due to testing. The effects of alterations to ω can be seen in figure 4.13.

(a) Input, and Dehazed Output with omega = 0

(b) Dehazed Output with omega = 1.0

(c) Dehazed Output with omega = 0.9

(d) Dehazed Output with omega = 0.75

Figure 4.13.: Figure shows the effects of decreased ω to the output image.

In order to demonstrate the trade-off between image quality and degree of dehazing, when changing the parameters, the following graphs are presented. The data relies on experiences earned during the parameter tests and are not exact measurements. The graph in figure 4.14 shows the effects of altered patchsize to the degree of dehazing and image quality. Figures 4.15 and 4.16 show the influence of the lower transmission bound and ω, respectively, to the image quality and degree of dehazing.

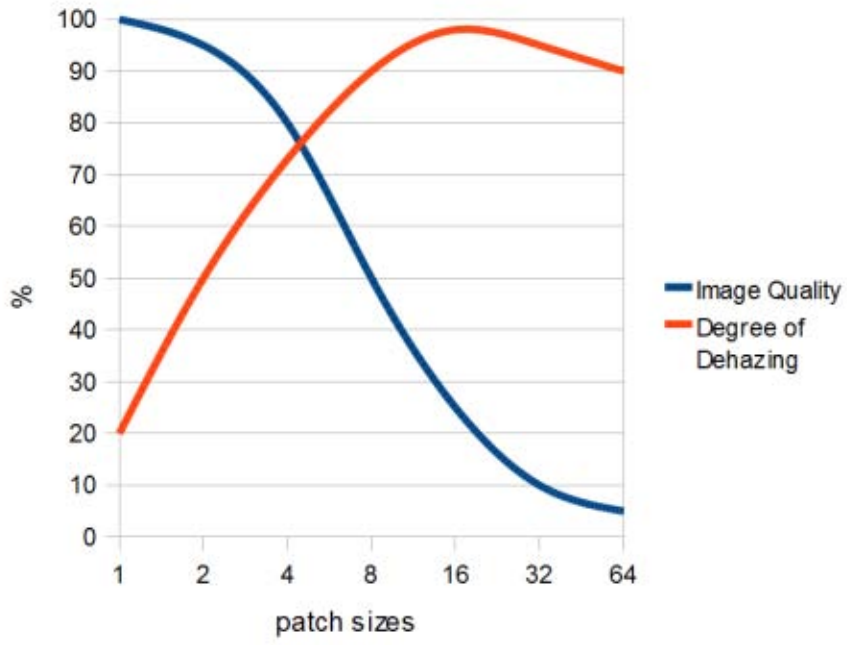


Figure 4.14.: Graph shows the empirical effects to image quality and degree of dehazing, when altering the patch sizes.

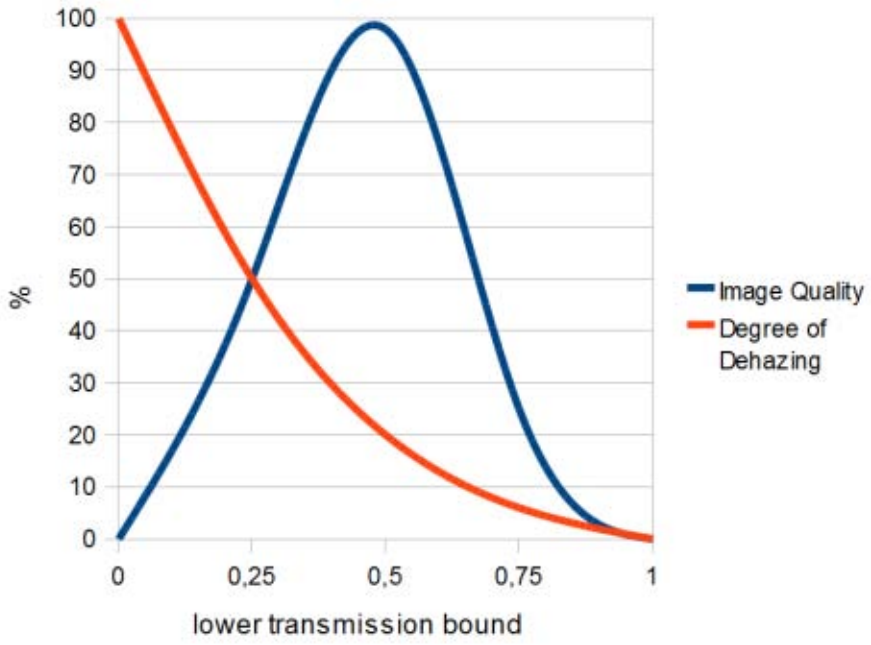


Figure 4.15.: Graph shows the empirical effects to image quality and degree of dehazing, when altering the lower transmission bound.

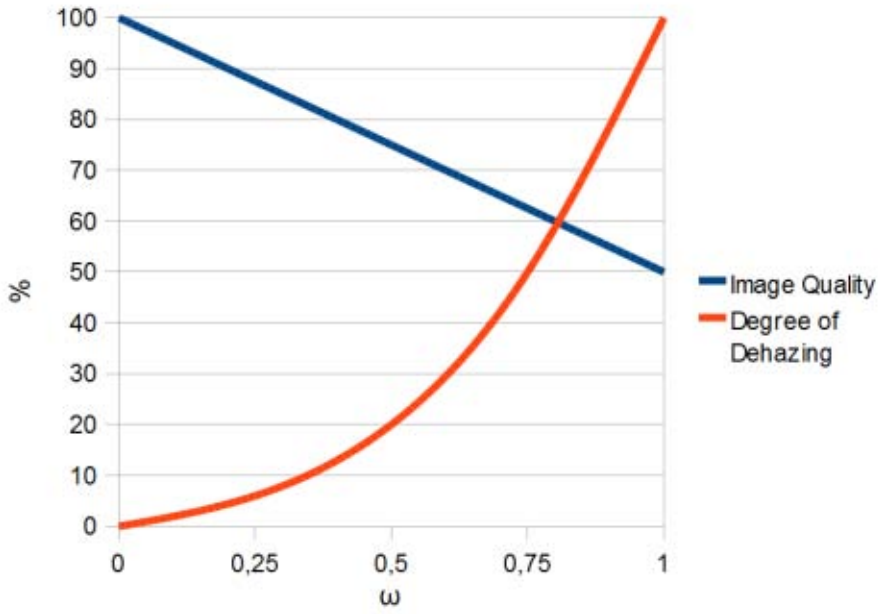


Figure 4.16.: Graph shows the empirical effects to image quality and degree of dehazing, when altering ω.

4.5. Further Research

The theory of the dark channel prior is physically sound and implementations like that of He et al. improve visibility in dehazed images. Other versions of implementation, like the one presented here, can perform dehazing fast, meaning, in real-time. However, this method still needs further research and improvements. Due to the eliminated soft matting step in this implementation, image quality is compromised in some cases as it has been shown. It is therefore desirable to find a filter for the transmission map that gives similar results as the soft matting, but can be performed in less time. Further Research could go into the direction of finding new tranmsission map filters with lower complexity, e.g. $\mathcal{O}(n)$. A comparison of the described methods in terms of complexity is given in the next chapter.

Like Kopf et al.'s research, more effort could also go into finding more applications for the depth map of the scene which could be easily attained from the transmission map. As another example for use, a new application was investigated by Hautière and Aubert [Hautière and Aubert, 2005] who dealt with on board cameras on street vehicles

who also performed dehazing/defogging, in this case it is obviously desirable to update the transmission map on every frame since the scene may change significantly on every frame, when at the same time high frame rates are required. Dehazing may also be used as a first image improvement step for other computer vision applications as well, like automatic object detection and tracking. In fact, many applications can benefit from dehazed images.

From the theoretical preliminary considerations(section 2.3, page 27) it is known that scattering is affected by wavelength, therefore the transmission depends on the pixel colour and it may be an improvement to generate one transmission map for each colour channel individually, as stated by [Lv et al., 2010].

As one can see, image dehazing is available in many different facets and for many possible applications, however one question remains, which method gives the best results and which method is the best for a specific application, respectively? The following chapter will deal with this subject in more detail.

5. Evaluation

This chapter is concerned with the evaluation of the developed dehazing method. This is an important step, because as more applications for dehazing may be found, and more dehazing algorithms may be developed, the question of which dehazing method gives the best results has to be asked eventually. For comparison reasons, a measure for both image quality and degree of dehazing have to be found. These measures don't exist, at least none that satisfy common consensus. In this chapter a measure is first introduced for the degree of dehazing, followed by two sections about measures for image quality and measures for speed. Then the specific test environment is described which is used for the evaluation with different weather scenarios in the next section. The last section of this chapter is concerned with possible sources of errors.

5.1. Measure for Degree of Dehazing

The author would like to take the opportunity and introduce a measure for the degree of dehazing, since this does not seem to be done by other researchers so far. The basic idea behind this measure is to ascertain the visual range in the input and output image, respectively and then calculating the quotient R_D, this quotient shall be called *Dehazing Range Quotient*:

$$R_D = \frac{R_o}{R_i}. \tag{5.1}$$

With R_o being the visual range in the dehazed output image and R_i the visual range in the input image. Referring to the *Runway Visual Range*, as introduced on page 14, the visual range may be acquired from the runway center line markings. However, in

general gradually increasing marks in the image may be used to determine the furthest just distinguishable mark. The distance to that mark may be calculated from GPS positions(and cross referenced e.g. due to satellite imaging), if the exact position of the camera is known. If the camera position is unknown, then counting the marks that are visible up to a certain point should fulfill the requirements as well. Assuming that the marks have the same distance to each other and the distance of the marks to the camera increases with each mark. This is a very simple measure, however already proven to work in meteorology. The quotient does not give an absolute value for dehazing degree, since it will always be relative to the input. The dehazing range quotient depends on several parameters of the test. For example, which medium or type of monitor is used to display the images, the test person and the surrounding light sources. Therefore the test for R_o and R_i should be done by the same test person under the same premises. Also, tests showed that the input image should be examined first, because it seems that psychological aspects interfere with the detection of small objects with low contrasts. As an example, although an object was initially not detected on the input, but on the output image, it was after looking at the input visible again since the test person knew that the object must be there. Additionally, it may be said that both images should be viewed at the same distance to the observer with the same level of zoom. As in the tiffany foundation tests, a certain probability of detection should be specified, too. In the case of center lines on airports, the runway lights, which are self illuminated and have the same distance each, are perfect for testing. To evaluate this dehazing method, the scene as shown in figure 5.1 is used.

(a) Input Image (b) Dehazed Output Image

Figure 5.1.: Figure shows the input image and the dehazed image of a scene used to determine the dehazing range quotient. The image was taken at Cologne airport under light fog condition.

Two tests were made with this pair of input and output images. First, the taxiway center line to the right was used, this center line contains integrated lights at a constant interval of 15m. As one can see in the above pictures, the lights are not visible as such, which already shows an improvement of the dehazed image over the original. In a first measure, the point was determined where the center line has the same luminance as the surrounding tarmac and is no longer distinguishable from the tarmac. Next, in the output, the number of visible lights on the center line were counted. To have the same unit for the dehazing range quotient, the images were cross referenced with a satellite image(Google Earth^{TM}) to determine the number of centerline lights that lay within the range determined for the input image. Now both images have a visual range expressed as number of visual center line lights along one line. The estimated quotient for this first test is:

$$\widetilde{R}_D = \frac{\widetilde{R}_o}{\widetilde{R}_i} \approx \frac{26}{12} \approx 2.2. \tag{5.2}$$

The cross reference with Google Earth^{TM}is shown in figure 5.2.

Figure 5.2.: Figure shows the cross reference of the test scene with a Google Earth^{TM}satellite image.

Due to the cross reference, real distances are available, using these values, the dehazing range quotient becomes:

$$R_D = \frac{R_o}{R_i} \approx \frac{498.1m}{292,5m} \approx 1.7. \tag{5.3}$$

This shows that the dehazed image indeed improved the visual range, since the quotient is greater than one. Different dehazing methods could be compared with the dehazing range quotient assuming the same input and the same units of measurement. Also noticeable in this example are several lights, such as those in the marked spot by the circle in figure 5.3, those are not at all visible for the bare eye in the input, but in the output. Unfortunately, those lights can't be used as proper marks for measuring since there are no other marks in between the camera and those lights. Thus, no exact quotient could be obtained from this, when marks with gradually increasing distance are missing. Instead, the lights marked by arrows were used for the measurement.

(a) Input Image (b) Dehazed Output Image

Figure 5.3.: Figure shows the marks used for visual range measurement in the test scene. Apparent in the output image are lights that are not visible in the input image.

The centerline light colour is a green, however a better mark would be a black body, since at daylight this kind of mark could be seen even further than a small source of light. Hence, a second test was performed using black bodies in the scene shown in the next figure, 5.4.

(a) Input Image (b) Dehazed Output Image

Figure 5.4.: Figure shows the scenes for visual range measurements using a black body.

The black bodies that were used are marked with red arrows in the above figure. Only the last visible black body is marked. When zoomed in very far, another even further black body becomes visible, this shall be the upper boundary of visual range in this scene. Cross referencing this scene with GPS data(figure 5.5) gives real distances which are used to determine the dehazing range quotient for this scene. The black body not visible in unzoomed view is marked with a yellow arrow in the image.

Figure 5.5.: Figure shows the cross reference of the test scene with a Google Earth^{TM}satellite image.

Since the marks are very far away from each other, only an interval can be given as certain:

$$1.0 < R_D = \frac{1953m}{1688m} (= 1.16) < \frac{R_max}{R_i} = \frac{2164m}{1688m} \tag{5.4}$$

$$1.0 < R_D < R_{D(max)} (= 1,28). \tag{5.5}$$

It may be convenient to present the quotient in a logarithmic scale, when the haze is only slight. Whereas for heavy fog, a linear scale seems to be more practical.

5.2. Measure for Image Quality

Defining the quality of an image is very complicated and highly influenced by psychological aspects that may vary from person to person. What defines a good image also depends on the application, if the application is, as here for example, to improve visi-

bility for faster detection of objects within a scene, then the proper way to evaluate it would be to make a field study with several test persons and measuring the probability of detection. This is of course out of the scope of this thesis and since there is no universal measure for image quality that has common consensus among computer vision scientists the only way to evaluate the images is to judge them by subjective means. Hence the judging of the image quality of the dehazer is up to the reader. An opportunity to do this is given in section 5.5.

5.3. Measure for Speed

In terms of processing speed, it was already stated that the developed method is able to process multiple images per second. However, direct comparison with other researcher's dehazers can not easily be performed on the basis of processing time, since a reference machine has to be used. Nevertheless, a comparison for the methods where procressing times are known, is given in table 5.1, as reference an image with size 600x400 is used.

Table 5.1.: Processing Times of Various Dehazing Methods

Method	Processing Time
Tan's Method	5-7 minutes
Fattal's Method	35 s
He et al.'s Method	10-20 s
Lv et al.'s GPU Method	80-100 ms
The Author's Method	30-50 ms

Note that the processing times got calculated with very different processing units from different hardware generations. More convenient therefore is the comparison in terms of complexity, which is independent of hardware, as opposed to absolute processing times. All modules in the author's method have a complexity of $\mathcal{O}(n)$, resulting in an overall complexity of also $\mathcal{O}(n)$ with n being the amount of pixels in the input image. However, when using a refined transmission map produced by, for example, Levin's Soft Matting algorithm, where the complexity is $\mathcal{O}(n^2)$, time consumption is too high for real-time applications, since the overall complexity becomes $\mathcal{O}(n^2)$, too. Other researchers already found faster ways, like Fang [Fang et al., 2010], who recently proposed an improved version of He et al.'s dark channel prior method, where the transmission map was determined

by first segmenting the input into similar areas with the same apparent surface, rather than dividing it in equally large patches and then calculating the transmission for the segment. Smoothing the transmission map according to Fang can then be done by a cross bilateral filter resulting in an overall complexity for the whole dehazing process of $\mathcal{O}(n \cdot \log n)$. Most of the methods of other researchers are of complexity $\mathcal{O}(n^2)$ or higher.

5.4. Test Environment

It was the desire of the author to reproduce a real world environment as best as possible. In a real world environment for air traffic ground surveillance video data is send through an Ethernet network with a framerate of 1 to 25 per second, depending on the model of camera and number of cameras. The video data usually consists of jpeg images, one for each frame, it needs to be decoded in order to access the rgb pixel values, which is necessary for the dehazing algorithm. With this in mind, the test environment was build in a way that a camera emulator software was used to simulate an IP-camera that sends out jpeg images with a specified frame rate over Ethernet. The camera simulator is fed with videos encoded with the xvid codec, for this software a separate PC was used, this PC shall be called "camera simulation PC" from here on. A detailed description of the camera simulation PC can be found in table 5.2. The simulation PC is connected through an Ethernet switch with a bandwidth of 1 Gbit/s to a second PC, from here on called the "dehazing PC". The job of the dehazing PC, as inferred by its name, is to receive the jpegs over Ethernet, decode and dehaze them. The dehazing software contains a jpeg decoding module utilising the Intel Jpeg Library(IJL), which is considered to be one of the fastest available. After decoding the jpeg, the dehazing algorithm described in the preceding section enhances the image. Afterwards, two images, both the dehazed and the original image is shown in an OpenGL video panel side by side. A detailed description of the dehazing pc can also be found in table 5.2. The described environment is visualised in figure 5.6.

Table 5.2.: Description of the Processing PCs in the Test Environment

Attribute	Camera Simulation PC	Dehazing PC
Operating System	MS Windows XP Professional SP3 32bit	MS Windows 7 Professional 64bit
CPU	AMD Opteron 150(Dual Core) @2Ghz	Intel Core i5 750(Quad Core) @2,67Ghz
Graphics Card	Nvidia GeForce 7800GT PCIe, 256MB GDDR3 RAM	Nvidia GeForce GTS 250, PCIe, 1GB GDDR3 RAM
Motherboard	Gigabyte GA-K8NF9 Ultra	Asus P7P55D Pro
Working Memory	2GB DDR(200Mhz) CL3	8GB DDR3(1333Mhz) CL8
Network	Nvidia Chip, Gigabit, Cat6 Cable	Realtek Chip, Gigabit, Cat6 Cable
Software & Tools	Searidge Technologies CamSim 0.8.9.7	Dehazing Software

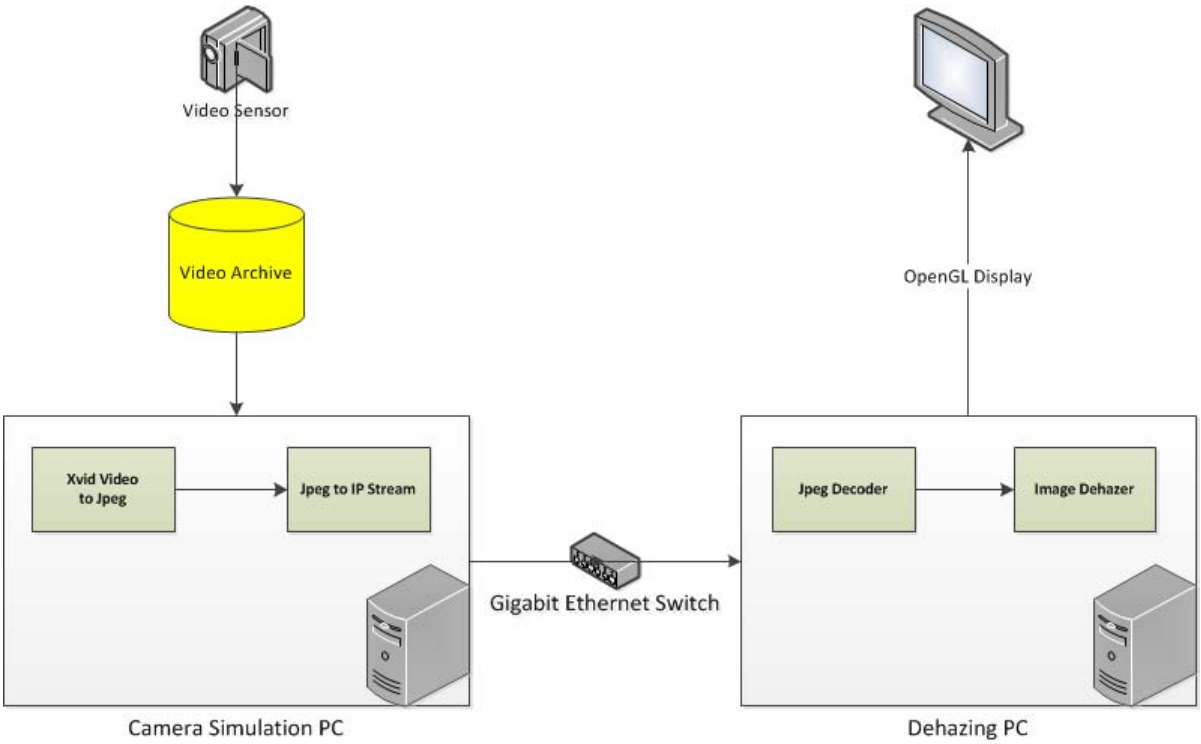


Figure 5.6.: Figure shows a schematic of the test environment.

5.5. Weather Scenarios

Several weather scenarios were tested with the developed dehazer. For each scenario there is a recorded video of the working dehazer on the CD-ROM which was attached to the thesis. The videos can also be reached through the links that are mentioned in the subsequent subsections. Those YouTube™videos are not listed and therefore not locatable via the common YouTube™search. An additional video for a grey scale infrared light scenario with thick fog can be found on the CD-ROM, too (and here: `http://www.youtube.com/watch?v=T77Ncl6g01Y`). The videos show the input image in the top left, the output in the top right and the transmission map on the bottom left of the video. All parameters and evaluation metrics are shown in the bottom right of the video. The CPU usage during the tests was at about 30-40% for each core, depending on the size of the input image and the used patchsize. For the tests here, gamma correction was used (no histogram equalisation) and only a simple transmission map (no soft matting or other filtering) to guarantee high framerates.

5.5.1. Clear Weather

As a reference a clear weather scenario was tested to show that the dehazing does not lower the quality of an already haze free image. Since there is no big effect in the output this test is only represented by figure 5.7.

(a) Input Image (b) Dehazed Output Image

Figure 5.7.: Scene with clear weather at Malta International Airport.

The dehazer does not distort the image, however overestimates the tarmac due to its bright colour, which results in an over saturation of the tarmac surfaces. The outlines of objects appear more clear due to the unrefined transmission map (patchsize 4x4). In the top left, the sky colour appears also stronger, here the sunlight directly hitting the camera lens causes the CCD sensor to create unnatural colours, which gets amplified by the dehazing algorithm.

5.5.2. Light Haze

This scenario tests the dehazing capabilities when a light haze is present in the scene. Figure 5.8 shows the scenario, videos can be obtained online(link: `http://www.youtube.com/watch?v=2BQHCyC1Q0c`) and from the CD-ROM.

(a) Input Image (b) Dehazed Output Image

Figure 5.8.: Scene with a light haze at Abu Dhabi International Airport.

As expected, haze is removed easily without introducing any errors to the image, since the dehazer was build with haze as such in mind, unlike the following scenarios. In the output, scene radiance is restored and objects are distinguishable more easily.

5.5.3. Light Fog

Next, a light fog scenario is tested. The scenario is represented by figure 5.9 and the light fog video file on the CD-ROM(also reachable through this link: `http://www.youtube.com/watch?v=jhZBer8vucc`).

(a) Input Image (b) Dehazed Output Image

Figure 5.9.: Scene with light fog at Cologne Airport.

Colours appear more vivid and the haze layer seems to be removed from the image.

5.5.4. Heavy Fog

Here the dehazing algorithm seem to reach its limitations, if there is no information in the input, then no haze layer may be removed and no original colour restored. The input shows heavy fog in the line of sight between the video sensor and the sun, the illumination of the fog particles towards the sun causes the sensor to get flooded by light. This results in huge areas where the pixel value is constantly near maximum white. The scenario can be seen in the heavy fog video file from the CD-ROM(link: `http://www.youtube.com/watch?v=f1XN_XQzN-A`) and in figure 5.10.

(a) Input Image (b) Dehazed Output Image

Figure 5.10.: Scene with heavy fog at Ottawa International Airport.

If the sun would not be radiating directly into the image sensor, then the fog could be removed more effectively. However in this case, only the parts of the image where the fog is not so heavily illuminated by the sun, seem to be improved in terms of haze layer removal and colour.

5.5.5. Heavy Rain

The next tested scenario is a heavy rain, additional difficulty stems from the fact that raindrops are directly on the lens. The scenario is shown in figure 5.11, as well as on the CD-ROM and online(link: `http://www.youtube.com/watch?v=7uwwpN9L8f4`).

(a) Input Image (b) Dehazed Output Image

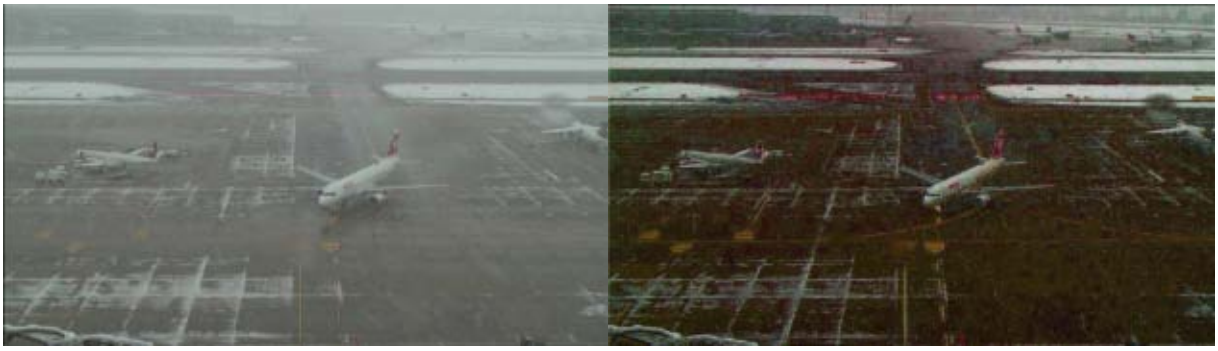
Figure 5.11.: Scene with heavy rain at Cologne Airport.

Although the visual range does not seem to have increased much, the improvements are

still visible, for example at the green center line lights, which are more apparent in the output. However, this scenario seems to get the least benefit from the dehazer from all tested scenarios here. This scene is also very dark, so it might be advantageous to apply a histogram equalisation instead of a gamma correction to the output.

5.5.6. Heavy Snowstorm

This scenario is the one most profiting from the dehazer, since the weather conditions are very bad in terms of visibility. The dehazer manages to improve visibility astonishingly well. Figure 5.12 shows the scene, the videos are available online(link: `http://www.youtube.com/watch?v=CnfdV-jZHnc`) and on the CD-ROM.



(a) Input Image (b) Dehazed Output Image

Figure 5.12.: Scene with heavy snowstorm at Zurich Airport.

Although the visibility is very low and snow is directly on the lens, colours are far more vivid than in the input and the occultation by the snow can be removed to a high degree. Especially in the top right of the image, planes become far more visible in the dehazed image. This was a good example for how well the dehazer works, other scenarios may have been less satisfying, to clarify the reasons for this possible sources of errors it is dealt with next.

5.6. Sources of Errors

Like in all signal processing, errors may occur due to noise and other factors. Computer vision algorithms are no exception, henceforth this subsection will discuss some of the errors that may have been occurred to the scenario testings. The error sources are divided into categories here, each is given its own subsection.

5.6.1. Errors Present in the Input

First are several errors that may already stem from the input, one of the most apparent errors are the block effects due to the jpeg compression in the presented scenarios. These block effects can get amplified by the dehazing step. In direct comparison with other works in the field of dehazing, the images presented here look much less satisfying, are however more realistic for real-time applications. Uncompressed images would of course look better after dehazing, but need far more bandwidth when sending them over Ethernet. Besides block effects, the quality when it comes to detail suffers from jpeg compression, since it has problems with high frequencies in the image. Assuming the same input the here developed dehazer would predictably perform similar in terms of image quality and detail. And of course noise is a big factor, that is already present in the input, especially far objects are prone to noise, "[..]noise in the restored images depends on distance. Far objects suffer from noise much more than close objects." [Schechner and Averbuch, 2007]. One possible solution is to use a noise suppression as suggested by [Schechner and Averbuch, 2007].

Also, limitations come from the image sensors that produce the input for the dehazer, image sensors like CCD and CMOS cameras are very sensitive devices that have a specific spectral response and sensitivity to illumination, all sensors used for the tests in this thesis are CCD cameras, they work well even in low light scenarios but are prone to noise more than their CMOS counterparts. And although these devices are similar to the working principle of the human eye, they can not reproduce scenes like they would be seen with the bare eye.

5.6.2. Conceptual Reasons

Secondly, there are errors that may be due to the concept of this method. The way that the atmospheric light is estimated, it may be possible that the whole image get a tint or colour cast. For example when very bright non white light sources are present in the scene that are even brighter than the sky. In this specific implementation there are also image quality demotions due to the missing transmission map filtering. There is always a trade-off between accuracy and speed. And then there is of course the general prior of the dark channel assumption that may be not fulfilled in some cases. For example when there are no shadows casted on large areas of the image and surfaces are relatively bright.

5.6.3. Theoretical Assumptions

Lastly, since Koschmieder's theory is no law, but a theory there may be several factors that could compromise the dehazing algorithm. The most obvious reason could be an unfulfilled requirement, as described in the chapter about Koschmieder's "Theorie der horizontalen Sichtweite" on page 45. Also those requirements for the scattering theories may be not met, like the assumptions that all particles are spherical. Some researcher believe, that haze removal alone can not produce sharp good images, like [Joshi and Cohen, 2010], who said: "[..](a) combination of shot noise and quantization noise is exacerbated when the contrast is expanded after haze removal. Dust on the sensor that may be unnoticeable in the original images creates serious artifacts.", therefore more steps before the haze removal would have to be performed. However, these possible sources of errors are theoretical, and since the videos demonstrate very good results in the test cases, the error that is introduced by these theoretical simplifications appears to be negligable under most conditions.

6. Discussions and Conclusions

In this work, the physical basis for dehazing algorithms has been exhibited and existing methods been described. Today's methods are physically sound and produce qualitatively good results, however for real-time applications they may not always be fast enough. For this thesis a new method was developed on the basis of the dark channel prior, where the complexity could be reduced from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$. New to this method as opposed to other methods, is the fact that it works without transmission map filtering when the patch sizes are sufficiently small. Also an improvement is the parallelisation that was made possible only by an unrefined transmission map. New filter techniques have been tested by the author too, but not found to be a sufficiently good improvement to the resulting images, at least not in terms of computational complexity justification. For the evaluation an extensive environment was implemented by the author that reproduces a real world application in air traffic ground surveillance, as shown in figure 5.6(page 107). Experimental results show the validity and effectiveness of this method with the help of the implemented test environment. It is shown that the dehazing method is capable of performing the task in a real-time fashion on modern standard computer hardware. For the sake of application orientation, many sets of parameters have been tested by the author that are the best usuable for air traffic ground surveillance scenarios. A new measure for the degree of dehazing has been proposed, the *Dehazing Range Quotient*. That enables for the first time the direct comparison of various dehazing methods. It has been shown in the tests here that although the method was built with just haze in mind, it is possible to even improve foggy, snowy and rainy scenes. Therefore, possible applications are broad, like outdoor surveillance or on board cameras in vehicles [Hautière and Aubert, 2005]. Some researchers even showed that it is possible to improve underwater imaging and aerial photography with similar, if not the same techniques [Schechner and Karpel, 2004b]. Tests in this thesis have confirmed that, like [Kopf et al., 2008] and [He et al., 2010a] stated in their papers, it may be advantageous not to remove the entire haze in the pictures since they may appear unnatural

and the observer may loose the awareness of scene depth, since humans rely on haze as an indicator for distance. However, it is certainly dependent on the type of application whether haze should be removed entirely or only to a certain degree.

A. Appendix - Dehazing Functions

A.1. Dehaze

A.1.1. Dehaze()

```
1  void CGLDialog::Dehaze()
2  {
3          if (m_dehazed) //only dehaze each frame once
4                  return;
5
6          DWORD dehazeStartTime = ::GetTickCount();
7
8          CSingleLock lock(&m_camBuffer->GetCriticalSection());
9          CSingleLock lockDehazer(&m_criticalSectionDehazedImage);
10
11         lock.Lock();
12         lockDehazer.Lock();
13
14         if(!((m_camBuffer->getHeight() % m_patchsizes[m_patchsize] == 0)
               && (m_camBuffer->getWidth() % m_patchsizes[m_patchsize] == 0)))
15                 return;
16
17         int size = m_camBuffer->getHeight() * m_camBuffer->getWidth() * 3;
18
19         //HistorgramSplayHSV(m_camBuffer->getBuffer(), size);
20
21         m_darkChannelCalcTime = 0;
22         m_transmissionCalcTime = 0;
23         m_brightnessCorrectionTime = 0;
24         m_recoverTime = 0;
25         if(!m_complexTransmissionMap)
26         {
```

```
27                  DWORD starttime = ::GetTickCount();
28                  if (m_multiThreading)
29                          FindDarkChannelMultiThread(m_patchsizes[
                                m_patchsize], size);
30                  else
31                          FindDarkChannel(m_patchsizes[m_patchsize], size);
32                  m_darkChannelCalcTime = ::GetTickCount() - starttime;
33
34                  CalcAtmosphericLight(); //doesn't take significant
                        processing time, hence no permanent time measurement!
35
36                  starttime = ::GetTickCount();
37                  CreateTransmissionMap(size);
38                  //HistorgramSplayGrey(m_transmissionMap, size/3);
39                  //HistorgramEqualisationGrey(m_transmissionMap, size/3);
40                  m_transmissionCalcTime = ::GetTickCount() - starttime;
41          }
42          else
43                  CreateComplexTransmissionMap(size);
44
45          DWORD starttime = ::GetTickCount();
46          if (m_multiThreading)
47                  CreateOutputImageMultiThread(size);
48          else
49                  CreateOutputImage(size);
50          m_recoverTime = ::GetTickCount() - starttime;
51
52          starttime = ::GetTickCount();
53          CorrectGamma(size);
54          //HistorgramEqualisationGrey(size);
55          //HistorgramEqualisationHSV(size);
56          //HistorgramEqualisationRGB(size);
57          //HistorgramSplayHSV(m_dehazedImageData, size);
58          m_brightnessCorrectionTime = ::GetTickCount() - starttime;
59
60          lockDehazer.Unlock();
61          lock.Unlock();
62
63          m_dehazeTime = ::GetTickCount() - dehazeStartTime;
64
65          m_dehazed = true;
66  }
```

A.2. FindDarkChannel

A.2.1. FindDarkChannel(..)

```
1  void CGLDialog::FindDarkChannel(int patchsize, int size)
2  {
3          //find dark channels of Input image I
4          //while performing step one, keep track of top 0,1% brightest
               pixels of the dark channel
5
6          int patches = (m_camBuffer->getHeight()/patchsize) * (m_camBuffer
               ->getWidth()/patchsize);
7
8          int width = m_camBuffer->getWidth() * 3; //RGB Channels
9
10         //create atmospheric light queue
11         int maxlength = (m_camBuffer->getHeight() * m_camBuffer->getWidth
               ()) / 1000;
12         m_queue = new atmosphericLightQueue(maxlength); //holds 0.1% of
               the brightest pixels in the dark channel
13
14         int patches_per_row = (width/3)/patchsize;
15
16         //loop over all patches
17         for (int patch=0; patch < patches; patch++)
18         {
19                 //find darkest colour channel in current patch:
20                 int darkest = 256;
21                 int darkest_x = -1;
22                 int darkest_y = -1;
23                 for (int n=0; n < patchsize; n++) //line
24                 {
25                         for (int m=0; m < patchsize*3; m+=3) //row
26                         {
27                                 int patch_row = patch / patches_per_row;
28                                 int patch_column = patch % patches_per_row
                                       ;
29                                 int x = m + patch_column*patchsize*3;
30                                 int y = n + patch_row*patchsize;
31                                 if(m_camBuffer->getBuffer()[y*width + x] <
                                       darkest)
32                                 {
```

```
33                                  darkest = m_camBuffer->getBuffer()
                                        [y*width + x];
34                                  darkest_x = x;
35                                  darkest_y = y;
36                              }
37                              if(m_camBuffer->getBuffer()[y*width + x +
                                    1] < darkest)
38                              {
39                                  darkest = m_camBuffer->getBuffer()
                                        [y*width + x + 1];
40                                  darkest_x = x;
41                                  darkest_y = y;
42                              }
43                              if(m_camBuffer->getBuffer()[y*width + x +
                                    2] < darkest)
44                              {
45                                  darkest = m_camBuffer->getBuffer()
                                        [y*width + x + 2];
46                                  darkest_x = x;
47                                  darkest_y = y;
48                              }
49                          }
50                      }
51
52                      //memorise the pixel that produced the darkchannel in the
                            patch and put it in the list of brightest pixels in the
                             dark channel
53                      if (darkest_x > -1 && darkest_y > -1)
54                              m_queue->insertValue(darkest,
55                               m_camBuffer->getBuffer()[darkest_y*width +
                                    darkest_x],
56                               m_camBuffer->getBuffer()[darkest_y*width +
                                    darkest_x + 1],
57                               m_camBuffer->getBuffer()[darkest_y*width +
                                    darkest_x + 2]
58                              );
59
60                      //set darkest channel as patch colour
61                      for (int n=0; n < patchsize; n++) //line
62                      {
63                              for (int m=0; m < patchsize*3; m+=3) //row
64                              {
```

```
65                                  int patch_row = patch / patches_per_row;
66                                  int patch_column = patch % patches_per_row
                                        ;
67                                  int x = m + patch_column*patchsize*3;
68                                  int y = n + patch_row*patchsize;
69
70                                  //set current darkest channel as patch
                                        colour
71                                  m_dehazedImageData[y*width + x] = darkest;
72                                  m_dehazedImageData[y*width + x + 1] =
                                        darkest;
73                                  m_dehazedImageData[y*width + x + 2] =
                                        darkest;
74                          }
75                  }
76          }
77  }
```

A.2.2. CalcAllPossiblePatchsizes()

```
1  //init function, not called at runtime
2  void CGLDialog::CalcAllPossiblePatchsizes()
3  {
4          int a = m_camBuffer->getWidth();
5          int b = m_camBuffer->getHeight();
6
7          if (b>a)
8          {
9                  int tmp = b;
10                 b = a;
11                 a = tmp;
12         }
13         for(int i = 1; i<=a; i++)
14         {
15                 if ((a%i == 0) && (b%i == 0))
16                         m_divisors++;
17         }
18
19         m_patchsizes = new int[m_divisors];
20         int counter = 0;
21
22         for(int i = 1; i<=a; i++)
```

```
23          {
24                  if ((a%i == 0) && (b%i == 0))
25                          m_patchsizes[counter++] = i;
26          }
27
28          return;
29  }
```

A.2.3. FindDarkChannelMultiThread(..)

```
1   //input image must be geater than 1000 pixels total
2   void CGLDialog::FindDarkChannelMultiThread(int patchsize, int size)
3   {
4           //find dark channels of Input image I
5           //while performing step one, keep track of top 0,1% brightest
                pixels of the dark channel
6
7           int patches = (m_camBuffer->getHeight()/patchsize) * (m_camBuffer
                ->getWidth()/patchsize);
8
9           int width = m_camBuffer->getWidth() * 3; //RGB Channels
10
11          //create atmospheric light queue
12          int maxlength = (m_camBuffer->getHeight() * m_camBuffer->getWidth
                ()) / 1000;
13          m_queue = new atmosphericLightQueue(maxlength); //holds 0.1% of
                the brightest pixels in the dark channel
14
15          //divide image into multiple sub-images
16          // 4 cores -> 4 subimages
17          int cores = 4;
18          int patchlines = m_camBuffer->getHeight()/patchsize;
19
20          int patchheight_all = patchlines / cores;
21          int patchheight_last;
22
23          if(patchlines % cores != 0)
24                  patchheight_last = patchlines / cores + (patchlines %
                        cores);
25          else
26                  patchheight_last = patchlines / cores;
27
```

```
28          int height_all  = patchheight_all*patchsize;
29          int height_last = patchheight_last*patchsize;
30
31          BYTE* subimage_1 = new BYTE[width*height_all];
32          memcpy(subimage_1, m_camBuffer->getBuffer(),  width*height_all);
33          BYTE* subimage_2 = new BYTE[width*height_all];
34          memcpy(subimage_2, m_camBuffer->getBuffer()+  width*height_all,
                width*height_all);
35          BYTE* subimage_3 = new BYTE[width*height_all];
36          memcpy(subimage_3, m_camBuffer->getBuffer()+2*width*height_all,
                width*height_all);
37          BYTE* subimage_4 = new BYTE[width*height_last];
38          memcpy(subimage_4, m_camBuffer->getBuffer()+3*width*height_all,
                width*height_last);
39
40          int patches_per_line = m_camBuffer->getWidth() / patchsize;
41          int stoppatch = patchheight_all*patches_per_line;
42
43          //call threads
44
45          unsigned int threadID1, threadID2, threadID3, threadID4;
46
47          mapThreadArgList arglist1;
48          arglist1.buf = subimage_1;
49          arglist1.size = patchsize;
50          arglist1.stoppatch = stoppatch;
51          arglist1.simplewidth = m_camBuffer->getWidth();
52          HANDLE threadHandle_1 = (HANDLE)_beginthreadex(NULL, 0, &CGLDialog
                ::CallThreadDarkChannel, (void*)&arglist1, 0, &threadID1);
53
54
55          mapThreadArgList arglist2;
56          arglist2.buf = subimage_2;
57          arglist2.size = patchsize;
58          arglist2.stoppatch = stoppatch;
59          arglist2.simplewidth = m_camBuffer->getWidth();
60          HANDLE threadHandle_2 = (HANDLE)_beginthreadex(NULL, 0, &CGLDialog
                ::CallThreadDarkChannel, (void*)&arglist2, 0, &threadID2);
61
62          mapThreadArgList arglist3;
63          arglist3.buf = subimage_3;
64          arglist3.size = patchsize;
```

```
65          arglist3.stoppatch = stoppatch;
66          arglist3.simplewidth = m_camBuffer->getWidth();
67          HANDLE threadHandle_3 = (HANDLE)_beginthreadex(NULL, 0, &CGLDialog
                ::CallThreadDarkChannel, (void*)&arglist3, 0, &threadID3);
68
69          stoppatch = patchheight_last*patches_per_line;
70          mapThreadArgList arglist4;
71          arglist4.buf = subimage_4;
72          arglist4.size = patchsize;
73          arglist4.stoppatch = stoppatch;
74          arglist4.simplewidth = m_camBuffer->getWidth();
75          HANDLE threadHandle_4 = (HANDLE)_beginthreadex(NULL, 0, &CGLDialog
                ::CallThreadDarkChannel, (void*)&arglist4, 0, &threadID4);
76
77          //wait for threads to be finished
78          WaitForSingleObject( threadHandle_1, INFINITE );
79          CloseHandle( threadHandle_1 );
80          WaitForSingleObject( threadHandle_2, INFINITE );
81          CloseHandle( threadHandle_2 );
82          WaitForSingleObject( threadHandle_3, INFINITE );
83          CloseHandle( threadHandle_3 );
84          WaitForSingleObject( threadHandle_4, INFINITE );
85          CloseHandle( threadHandle_4 );
86
87          //reassemble the subimages
88          memcpy(m_dehazedImageData,
                subimage_1, width*height_all);
89          memcpy(m_dehazedImageData+  width*height_all, subimage_2, width*
                height_all);
90          memcpy(m_dehazedImageData+2*width*height_all, subimage_3, width*
                height_all);
91          memcpy(m_dehazedImageData+3*width*height_all, subimage_4, width*
                height_last);
92
93          delete[] subimage_1;
94          delete[] subimage_2;
95          delete[] subimage_3;
96          delete[] subimage_4;
97  }
```

A.2.4. CallThreadDarkChannel(..)

```
1  unsigned _stdcall CGLDialog::CallThreadDarkChannel(void* args)
2  {
3          mapThreadArgList argList = *((mapThreadArgList*)args);
4          FindDarkChannel(argList.buf, argList.stoppatch, argList.size,
               argList.simplewidth);
5          return 1;
6  }
```

A.2.5. FindDarkChannel(..) in Subimage

```
1  void CGLDialog::FindDarkChannel(BYTE* buf, int stoppatch, int patchsize,
       int simplewidth)
2  {
3          int width = simplewidth * 3;
4          int patches_per_row = simplewidth/patchsize;
5          //loop over all patches
6          for (int patch=0; patch < stoppatch; patch++)
7          {
8                  //find darkest colour channel in current patch:
9                  int darkest = 256;
10                 int darkest_x = -1;
11                 int darkest_y = -1;
12                 for (int n=0; n < patchsize; n++) //line
13                 {
14                         for (int m=0; m < patchsize*3; m+=3) //row
15                         {
16                                 int patch_row = patch / patches_per_row;
17                                 int patch_column = patch % patches_per_row
                                       ;
18                                 int x = m + patch_column*patchsize*3;
19                                 int y = n + patch_row*patchsize;
20                                 if(buf[y*width + x] < darkest)
21                                 {
22                                         darkest = buf[y*width + x];
23                                         darkest_x = x;
24                                         darkest_y = y;
25                                 }
26                                 if(buf[y*width + x + 1] < darkest)
27                                 {
28                                         darkest = buf[y*width + x + 1];
29                                         darkest_x = x;
30                                         darkest_y = y;
```

```
31                              }
32                              if(buf[y*width + x + 2] < darkest)
33                              {
34                                      darkest = buf[y*width + x + 2];
35                                      darkest_x = x;
36                                      darkest_y = y;
37                              }
38                      }
39              }
40
41              //memorise the pixel that produced the darkchannel in the
                    patch and put it in the list of brightest pixels in the
                     dark channel
42              CSingleLock lock(&theApp.m_pGLDialog->
                    m_criticalSectionAtmosphericLightQueue);
43
44              lock.Lock();
45
46              if (darkest_x > -1 && darkest_y > -1)
47                      theApp.m_pGLDialog->m_queue->insertValue(darkest,
48                                                              buf[
                                                                  darkest_y
                                                                  *width
                                                                  +
                                                                  darkest_x
                                                                  ],
49                                                              buf[
                                                                  darkest_y
                                                                  *width
                                                                  +
                                                                  darkest_x
                                                                   + 1],
50                                                              buf[
                                                                  darkest_y
                                                                  *width
                                                                  +
                                                                  darkest_x
                                                                   + 2]);
51
52              lock.Unlock();
53
54              //set darkest channel as patch colour
```

```
55                  for (int n=0; n < patchsize; n++) //line
56                  {
57                          for (int m=0; m < patchsize*3; m+=3) //row
58                          {
59                                  int patch_row = patch / patches_per_row;
60                                  int patch_column = patch % patches_per_row
                                        ;
61                                  int x = m + patch_column*patchsize*3;
62                                  int y = n + patch_row*patchsize;
63
64                                  //set current darkest channel as patch
                                        colour
65                                  buf[y*width + x] = darkest;
66                                  buf[y*width + x + 1] = darkest;
67                                  buf[y*width + x + 2] = darkest;
68                          }
69                  }
70          }
71  }
```

A.3. CreateTransmissionMap

A.3.1. CreateTransmissionMap(..)

```
1   void CGLDialog::CreateTransmissionMap(int size)
2   {
3           //get the highest channel of the atmospheric light
4           //goal is to minimise m_dehazedImageData[x] / A_max
5
6           double A_max = m_atmosphericLight[0];
7           if (m_atmosphericLight[1] > A_max)
8                   A_max = m_atmosphericLight[1];
9           if (m_atmosphericLight[2] > A_max)
10                  A_max = m_atmosphericLight[2];
11
12          //calc the transmission
13          for (int x=0; x < size; x+=3)
14          {
15                  float normalisedDarkChannel = ((double)m_dehazedImageData[
                        x] / A_max);
```

```
16
17                  m_transmissionMap[x/3] = 1.0f - m_levelOfRealism*
                        normalisedDarkChannel;
18
19                  if(m_showTransmissionTexture)
20                          m_transmissionTexture[x + 2] =
                                m_transmissionTexture[x + 1] =
                                m_transmissionTexture[x] = (int)(
                                m_transmissionMap[x/3] * 255.0f);
21          }
22  }
```

A.3.2. CreateComplexTransmissionMap(..)

```
1   void CGLDialog::CreateComplexTransmissionMap(int size)
2   {
3           if(m_isTransmissionMapCreated)
4                   return;
5
6           FindDarkChannel(m_patchsizes[m_patchsize], size);
7           CalcAtmosphericLight();
8
9           double A_max = m_atmosphericLight[0];
10          if (m_atmosphericLight[1] > A_max)
11                  A_max = m_atmosphericLight[1];
12          if (m_atmosphericLight[2] > A_max)
13                  A_max = m_atmosphericLight[2];
14
15          //calc the transmission
16          for (int x=0; x < size; x+=3)
17          {
18                  double normalisedDarkChannel = ((double)m_dehazedImageData
                        [x] / A_max);
19
20                  m_transmissionMap[x/3] = 1.0f - m_levelOfRealism*
                        normalisedDarkChannel;
21          }
22
23          FilterTransmissionMap(size);
24
25          m_isTransmissionMapCreated = true;
26
```

```
27          if(m_showTransmissionTexture)
28          for (int x=0; x < size; x+=3)
29          {
30                  m_transmissionTexture[x + 2] = m_transmissionTexture[x +
                        1] = m_transmissionTexture[x] = (int)(m_transmissionMap
                        [x/3] * 255.0f);
31          }
32  }
```

A.3.3. FilterTransmissionMap(..)

```
1   void CGLDialog::FilterTransmissionMap(int size)
2   {
3           Blur2TransmissionMaps(8, size);
4           //GaussBlurTransmissionMap(5, size);
5   }
```

A.3.4. Blur2TransmissionMaps(..)

```
1   void CGLDialog::Blur2TransmissionMaps(int patchsize, int size)
2   {
3           FindDarkChannel(patchsize, size);
4           CalcAtmosphericLight();
5
6           double* tmp_trans = new double[size/3];
7
8           double A_max = m_atmosphericLight[0];
9           if (m_atmosphericLight[1] > A_max)
10                  A_max = m_atmosphericLight[1];
11          if (m_atmosphericLight[2] > A_max)
12                  A_max = m_atmosphericLight[2];
13
14          //calc the transmission of second transmission map
15          for (int x=0; x < size; x+=3)
16          {
17                  double normalisedDarkChannel = ((double)m_dehazedImageData
                        [x] / A_max);
18
19                  tmp_trans[x/3] = 1.0f - m_levelOfRealism*
                        normalisedDarkChannel;
20          }
```

```
21
22          double weight_1 = 0.95;
23          double weight_big = 0.05;
24
25          //blur the two transmission maps
26          for (int i=0; i < size/3; i++)
27          {
28                  m_transmissionMap[i] = tmp_trans[i]*weight_big +
                        m_transmissionMap[i]*weight_1;
29          }
30
31          delete[] tmp_trans;
32  }
```

A.3.5. GaussBlurTransmissionMap(..)

```
1   void CGLDialog::GaussBlurTransmissionMap(int k, int size)
2   {
3           double* tmp_trans = new double[size/3];
4
5           double** ker = new double*[2*k+1];
6           for (int i = 0; i < 2*k+1; i++ )
7                   ker[i] = new double[2*k+1];
8
9           //init kernel
10          for (int i=0; i<2*k+1; i++)
11                  for (int a=0; a<2*k+1; a++)
12                  {
13                          double max_rad = 4*k*k;
14                          double sum = (2*k+1)*(2*k+1);
15                          double nonNull_cases = (2*k+1)*(2*k+1)-4;
16                          double x = a-k;
17                          double y = i-k;
18                          double rad = sqrt(pow(x,2) + pow(y,2));
19                          ker[i][a] = (max_rad - rad)/(max_rad*nonNull_cases
                                );
20                  }
21
22          //do the blurring
23          for (int y=0;y<m_camBuffer->getHeight();y++)
24          {
25                  for (int x=0;x<m_camBuffer->getWidth();x++)
```

```
26                  {
27                          float total=0.0;
28                          for (int j=-k;j<=k;j++)
29                          {
30                                  for (int i=-k;i<=k;i++)
31                                  {
32                                          int x2=x+i;
33                                          int y2=y+j;
34                                          if (x2>=0 && x2<m_camBuffer->
                                                getWidth() && y2>=0 && y2<
                                                m_camBuffer->getHeight())
35                                                  total+=ker[j+k][i+k]*
                                                        m_transmissionMap[y2*
                                                        m_camBuffer->getWidth()
                                                        +x2];
36                                  }
37                          }
38                          if (total<0.0) total=0.0; else if (total>1.0)
                                total=1.0;
39                          tmp_trans[x+y*m_camBuffer->getWidth()]=total;
40                  }
41          }
42
43          int size_t = 8*size/3;
44          memcpy(m_transmissionMap, tmp_trans, size_t);
45          delete[] tmp_trans;
46  }
```

A.4. CalcAtmosphericLight

A.4.1. CalcAtmosphericLight()

```
1  void CGLDialog::CalcAtmosphericLight()
2  {
3          m_queue->calcAtmosphericLight();
4          m_atmosphericLight[0] = m_queue->getAtmosphericLightR();
5          m_atmosphericLight[1] = m_queue->getAtmosphericLightG();
6          m_atmosphericLight[2] = m_queue->getAtmosphericLightB();
7
8          m_queue->clearList();
```

```
9           delete m_queue;
10          m_queue = NULL;
11  }
```

A.5. CreateOutputImage

A.5.1. CreateOutputImage(..)

```
1
2   void CGLDialog::CreateOutputImage(int size)
3   {
4           //put all pieces together to the dehazed output image
5           //this is the realisation of formula 3.20 or in He's paper: 16,
                respectively
6
7           for (int x=0; x < size; x+=3)
8           {
9                   //decide what transmission to take (noise reduction)
10                  double tmp_transmission = m_transmissionMap[x/3];
11                  if (tmp_transmission < m_t0)
12                          tmp_transmission = m_t0;
13
14                  int pixelvalue_r = m_camBuffer->getBuffer()[x];
15                  if (pixelvalue_r > m_atmosphericLight[0])
16                          pixelvalue_r = m_atmosphericLight[0];
17
18                  int pixelvalue_g = m_camBuffer->getBuffer()[x+1];
19                  if (pixelvalue_g > m_atmosphericLight[1])
20                          pixelvalue_g = m_atmosphericLight[1];
21
22                  int pixelvalue_b = m_camBuffer->getBuffer()[x+2];
23                  if (pixelvalue_b > m_atmosphericLight[2])
24                          pixelvalue_b = m_atmosphericLight[2];
25
26                  m_dehazedImageData[x     ] = (((double)(pixelvalue_r -
                        m_atmosphericLight[0])) / tmp_transmission) +
                        m_atmosphericLight[0];
27
28                  m_dehazedImageData[x + 1] = (((double)(pixelvalue_g -
                        m_atmosphericLight[1])) / tmp_transmission) +
```

```
                    m_atmosphericLight[1];
29
30                  m_dehazedImageData[x + 2] = (((double)(pixelvalue_b -
                        m_atmosphericLight[2])) / tmp_transmission) +
                        m_atmosphericLight[2];
31          }
32  }
```

A.5.2. CreateOutputImageMultiThread(..)

```
1   void CGLDialog::CreateOutputImageMultiThread(int size)
2   {
3           //put all pieces together to the dehazed output image
4           //this is the realisation of formula 3.20 or in He's paper: 16,
                respectively
5
6           //divide image into multiple sub-images
7           // 4 cores -> 4 subimages
8           int cores = 4;
9
10          int height_all = m_camBuffer->getHeight() / cores;
11          int height_last;
12
13          int width = m_camBuffer->getWidth() * 3;
14
15          if(m_camBuffer->getHeight() % cores != 0)
16                  height_last = (m_camBuffer->getHeight() / cores) + (
                        m_camBuffer->getHeight() % cores);
17          else
18                  height_last = height_all;
19
20          //create subimages of input image
21          BYTE* subimage_1 = new BYTE[width*height_all];
22          memcpy(subimage_1, m_camBuffer->getBuffer()                    ,
                width*height_all);
23          BYTE* subimage_2 = new BYTE[width*height_all];
24          memcpy(subimage_2, m_camBuffer->getBuffer()+  width*height_all,
                width*height_all);
25          BYTE* subimage_3 = new BYTE[width*height_all];
26          memcpy(subimage_3, m_camBuffer->getBuffer()+2*width*height_all,
                width*height_all);
27          BYTE* subimage_4 = new BYTE[width*height_last];
```

```
28          memcpy(subimage_4, m_camBuffer->getBuffer()+3*width*height_all,
                width*height_last);
29
30          //create memory for dehazed subimages
31          BYTE* dehazedImageData_1 = new BYTE[width*height_all];
32          BYTE* dehazedImageData_2 = new BYTE[width*height_all];
33          BYTE* dehazedImageData_3 = new BYTE[width*height_all];
34          BYTE* dehazedImageData_4 = new BYTE[width*height_last];
35
36          //create memory for sub transmissionmap
37          double* transmission_1 = new double[(width/3)*height_all];
38          memcpy(transmission_1, m_transmissionMap                    ,
                8*(width/3)*height_all);
39          double* transmission_2 = new double[(width/3)*height_all];
40          memcpy(transmission_2, m_transmissionMap+  (width/3)*height_all,
                8*(width/3)*height_all);
41          double* transmission_3 = new double[(width/3)*height_all];
42          memcpy(transmission_3, m_transmissionMap+2*(width/3)*height_all,
                8*(width/3)*height_all);
43          double* transmission_4 = new double[(width/3)*height_last];
44          memcpy(transmission_4, m_transmissionMap+3*(width/3)*height_all,
                8*(width/3)*height_last);
45
46          //call threads
47
48          unsigned int threadID1, threadID2, threadID3, threadID4;
49
50          mapThreadArgList2 arglist1;
51          arglist1.buf_in = subimage_1;
52          arglist1.buf_out = dehazedImageData_1;
53          arglist1.size = width*height_all;
54          arglist1.light_r = m_atmosphericLight[0];
55          arglist1.light_g = m_atmosphericLight[1];
56          arglist1.light_b = m_atmosphericLight[2];
57          arglist1.t0 = m_t0;
58          arglist1.transmissionMap = transmission_1;
59          HANDLE threadHandle_1 = (HANDLE)_beginthreadex(NULL, 0, &CGLDialog
                ::CallThreadCalcOutput, (void*)&arglist1, 0, &threadID1);
60
61          mapThreadArgList2 arglist2;
62          arglist2.buf_in = subimage_2;
63          arglist2.buf_out = dehazedImageData_2;
```

```
64          arglist2.size = width*height_all;
65          arglist2.light_r = m_atmosphericLight[0];
66          arglist2.light_g = m_atmosphericLight[1];
67          arglist2.light_b = m_atmosphericLight[2];
68          arglist2.t0 = m_t0;
69          arglist2.transmissionMap = transmission_2;
70          HANDLE threadHandle_2 = (HANDLE)_beginthreadex(NULL, 0, &CGLDialog
                ::CallThreadCalcOutput, (void*)&arglist2, 0, &threadID2);
71
72          mapThreadArgList2 arglist3;
73          arglist3.buf_in = subimage_3;
74          arglist3.buf_out = dehazedImageData_3;
75          arglist3.size = width*height_all;
76          arglist3.light_r = m_atmosphericLight[0];
77          arglist3.light_g = m_atmosphericLight[1];
78          arglist3.light_b = m_atmosphericLight[2];
79          arglist3.t0 = m_t0;
80          arglist3.transmissionMap = transmission_3;
81          HANDLE threadHandle_3 = (HANDLE)_beginthreadex(NULL, 0, &CGLDialog
                ::CallThreadCalcOutput, (void*)&arglist3, 0, &threadID3);
82
83          mapThreadArgList2 arglist4;
84          arglist4.buf_in = subimage_4;
85          arglist4.buf_out = dehazedImageData_4;
86          arglist4.size = width*height_last;
87          arglist4.light_r = m_atmosphericLight[0];
88          arglist4.light_g = m_atmosphericLight[1];
89          arglist4.light_b = m_atmosphericLight[2];
90          arglist4.t0 = m_t0;
91          arglist4.transmissionMap = transmission_4;
92          HANDLE threadHandle_4 = (HANDLE)_beginthreadex(NULL, 0, &CGLDialog
                ::CallThreadCalcOutput, (void*)&arglist4, 0, &threadID4);
93
94          //wait for threads to be finished
95          WaitForSingleObject( threadHandle_1, INFINITE );
96          CloseHandle( threadHandle_1 );
97          WaitForSingleObject( threadHandle_2, INFINITE );
98          CloseHandle( threadHandle_2 );
99          WaitForSingleObject( threadHandle_3, INFINITE );
100         CloseHandle( threadHandle_3 );
101         WaitForSingleObject( threadHandle_4, INFINITE );
102         CloseHandle( threadHandle_4 );
```

```
103
104         //reassemble the subimages
105         memcpy(m_dehazedImageData,
                dehazedImageData_1, width*height_all);
106         memcpy(m_dehazedImageData+  width*height_all, dehazedImageData_2,
                width*height_all);
107         memcpy(m_dehazedImageData+2*width*height_all, dehazedImageData_3,
                width*height_all);
108         memcpy(m_dehazedImageData+3*width*height_all, dehazedImageData_4,
                width*height_last);
109
110         delete[] subimage_1;
111         delete[] subimage_2;
112         delete[] subimage_3;
113         delete[] subimage_4;
114         delete[] transmission_1;
115         delete[] transmission_2;
116         delete[] transmission_3;
117         delete[] transmission_4;
118         delete[] dehazedImageData_1;
119         delete[] dehazedImageData_2;
120         delete[] dehazedImageData_3;
121         delete[] dehazedImageData_4;
122 }
```

A.5.3. CallThreadCalcOutput(..)

```
1  unsigned _stdcall CGLDialog::CallThreadCalcOutput(void* args)
2  {
3         mapThreadArgList2 argList = *((mapThreadArgList2*)args);
4         CreateOutputSubImage(argList.buf_in, argList.buf_out, argList.size
              , argList.light_r, argList.light_g, argList.light_b, argList.t0
              , argList.transmissionMap);
5         return 1;
6  }
```

A.5.4. CreateOutputSubImage(..)

```
1  void CGLDialog::CreateOutputSubImage(BYTE* buf_in, BYTE* buf_out, int size
       , int light_r, int light_g, int light_b, double t0, double*
       transmissionMap)
```

```
2  {
3          for (int x=0; x < size; x+=3)
4          {
5                  //decide what transmission to take (noise reduction)
6                  double tmp_transmission = transmissionMap[x/3];
7                  if (tmp_transmission < t0)
8                          tmp_transmission = t0;
9
10                 int pixelvalue_r = buf_in[x];
11                 if (pixelvalue_r > light_r)
12                         pixelvalue_r = light_r;
13
14                 int pixelvalue_g = buf_in[x+1];
15                 if (pixelvalue_g > light_g)
16                         pixelvalue_g = light_g;
17
18                 int pixelvalue_b = buf_in[x+2];
19                 if (pixelvalue_b > light_b)
20                         pixelvalue_b = light_b;
21
22                 buf_out[x     ] = (((double)(pixelvalue_r - light_r)) /
                       tmp_transmission) + light_r;
23
24                 buf_out[x + 1] = (((double)(pixelvalue_g - light_g)) /
                       tmp_transmission) + light_g;
25
26                 buf_out[x + 2] = (((double)(pixelvalue_b - light_b)) /
                       tmp_transmission) + light_b;
27         }
28 }
```

B. Appendix - AtmosphericLightQueue Class

B.1. AtmosphericLightQueue.cpp

```
1  #include "StdAfx.h"
2  #include "atmosphericLightQueue.h"
3
4
5  //constructor
6  atmosphericLightQueue::atmosphericLightQueue(int length)
7  {
8          this->m_currentLength = 0;
9          this->m_maxLength = length; //must be greater than 1
10         this->m_last = NULL;
11         this->m_first = NULL;
12         this->m_atmosphericLightR = -1;
13         this->m_atmosphericLightG = -1;
14         this->m_atmosphericLightB = -1;
15 }
16
17 void atmosphericLightQueue::insertValue(int dark, int r, int g, int b)
18 {
19         if (this->m_currentLength < 0)
20                 return; //list not properly initialised
21
22         if (this->m_currentLength == 0) //if this is the first element to
               insert
23         {
24                 //create new list entry
25                 atmosphericLightElement* newElement = new
                       atmosphericLightElement(dark, r, g, b);
```

```
26                  this->m_currentLength++;
27                  this->m_first = this->m_last = newElement;
28          }
29          else //if this is not the first element to insert
30          {
31                  if ((this->m_currentLength >= this->m_maxLength) && (this
                        ->m_last->darkchannel >= dark)) //if list is full and
                        new element is darker than the darkest in list
32                          return; //dont insert element!
33
34                  //insert element
35                  //create new list entry
36                  atmosphericLightElement* newElement = new
                        atmosphericLightElement(dark, r, g, b);
37                  this->m_currentLength++;
38
39                  if (dark >= m_first->darkchannel) //newElement gets put
                        into first position
40                  {
41                          this->m_first->prev = newElement;
42                          newElement->next = this->m_first;
43                          this->m_first = newElement;
44
45                          //check if last element needs to be erased!
46                          if( this->m_currentLength > this->m_maxLength )
47                                  this->deleteDarkest();
48                          return;
49                  }
50
51                  if (dark <= this->m_last->darkchannel) //newElement gets
                        put into last position
52                  {
53                          this->m_last->next = newElement;
54                          newElement->prev = this->m_last;
55                          this->m_last = newElement;
56                          return;
57                  }
58
59                  //insert newElement in between
60                  atmosphericLightElement* tmpElement = NULL;
61                  for (tmpElement = this->m_first; tmpElement->next != NULL;
                         tmpElement = tmpElement->next)
```

```
62                  {
63                          if (tmpElement->darkchannel < dark)
64                                  break;
65                  } //this for loop navigates to the right position in the
                        list where the new element should be inserted
66
67                  //insert new element in front of tmpElement
68                  newElement->prev = tmpElement->prev;
69                  tmpElement->prev->next = newElement;
70                  tmpElement->prev = newElement;
71                  newElement->next = tmpElement;
72
73
74                  //check if last element needs to be erased!
75                  if( this->m_currentLength > this->m_maxLength )
76                          this->deleteDarkest();
77
78          }
79  }
80
81  //removes the last element in the list
82  bool atmosphericLightQueue::deleteDarkest()
83  {
84          //if (m_currentLength <= 0)
85          if (this->m_first == NULL)
86                  return false;
87
88          //if (m_currentLength == 1)
89          if (this->m_first == this->m_last)
90          {
91                  delete this->m_first;
92                  this->m_last = NULL;
93                  this->m_first = NULL;
94          }
95          else
96          {
97                  atmosphericLightElement* tmpElement;
98                  tmpElement = this->m_last->prev;
99                  tmpElement->next = NULL;
100                 delete this->m_last;
101                 this->m_last = tmpElement;
102         }
```

```
103
104         this->m_currentLength--;
105         return true;
106 }
107
108 //command to empty the list
109 void atmosphericLightQueue::clearList()
110 {
111         while (deleteDarkest()){}
112 }
113
114
115 void atmosphericLightQueue::printList()
116 {
117         if (m_currentLength <= 0)
118                 return;
119
120         int counter = 0;
121
122         for(atmosphericLightElement* tmpElement = this->m_first;
                tmpElement->next != NULL; tmpElement = tmpElement->next)
123         {
124                 TRACE(_T("%d"),tmpElement->darkchannel);
125                 TRACE(_T(","));
126                 counter++;
127         }
128         TRACE(_T("%d"),m_last->darkchannel);
129         TRACE(_T("| - %d"), ++counter);
130         TRACE(_T("!\n"));
131 }
132
133 void atmosphericLightQueue::printRGBList()
134 {
135         if (m_currentLength <= 0)
136                 return;
137
138         int counter = 0;
139
140         for(atmosphericLightElement* tmpElement = this->m_first;
                tmpElement->next != NULL; tmpElement = tmpElement->next)
141         {
```

```
142                     TRACE(_T("%d, %d, %d"),tmpElement->r_input, tmpElement->
                            g_input, tmpElement->b_input);
143                     TRACE(_T("| "));
144                     counter++;
145             }
146             TRACE(_T("%d, %d, %d"),m_last->r_input, m_last->g_input, m_last->
                    b_input);
147             TRACE(_T("| - %d"), ++counter);
148             TRACE(_T("!\n"));
149     }
150
151     int     atmosphericLightQueue::getLength()
152     {
153             return this->m_currentLength;
154     }
155
156
157     //for debugging
158     int atmosphericLightQueue::getRealLength()
159     {
160             if (m_currentLength <= 0)
161                     return 0;
162
163             int counter = 1;
164
165             for(atmosphericLightElement* tmpElement = this->m_first;
                    tmpElement->next != NULL; tmpElement = tmpElement->next)
166             {
167                     counter++;
168             }
169
170             return counter;
171     }
172
173     void atmosphericLightQueue::calcAtmosphericLight()
174     {
175             if (m_currentLength <= 0)
176             {
177                     this->m_atmosphericLightR = 0;
178                     this->m_atmosphericLightG = 0;
179                     this->m_atmosphericLightB = 0;
180                     return;
```

```
181         }
182
183         int temp_r = 0;
184         int temp_g = 0;
185         int temp_b = 0;
186
187         float temp_intensity = 0.0f;
188
189         //get element with highest intensity and set this pixel as
                atmospheric light
190         for(atmosphericLightElement* tmpElement = this->m_first;
                tmpElement != NULL; tmpElement = tmpElement->next)
191         {
192                 if (tmpElement->intensity_input > temp_intensity)
193                 {
194                         temp_intensity = tmpElement->intensity_input;
195                         temp_r = tmpElement->r_input;
196                         temp_g = tmpElement->g_input;
197                         temp_b = tmpElement->b_input;
198                 }
199         }
200
201         this->m_atmosphericLightR = temp_r;
202         this->m_atmosphericLightG = temp_g;
203         this->m_atmosphericLightB = temp_b;
204 }
205
206 int atmosphericLightQueue::getAtmosphericLightR()
207 {
208         return this->m_atmosphericLightR;
209 }
210
211 int atmosphericLightQueue::getAtmosphericLightG()
212 {
213         return this->m_atmosphericLightG;
214 }
215
216 int atmosphericLightQueue::getAtmosphericLightB()
217 {
218         return this->m_atmosphericLightB;
219 }
```

B.2. AtmosphericLightQueue.h

```
1  #pragma once
2
3  #include "atmosphericLightElement.h"
4
5  #ifndef ATMOSPHERICLIGHTQUEUE_H
6  #define ATMOSPHERICLIGHTQUEUE_H
7
8  class atmosphericLightQueue
9  {
10 private:
11         bool     deleteDarkest();
12
13         atmosphericLightElement*  m_last, m_first;
14
15         int              m_maxLength; //must be greater than 1
16         int              m_currentLength;
17
18         int      m_atmosphericLightR;
19         int      m_atmosphericLightG;
20         int      m_atmosphericLightB;
21
22 public:
23         //constructor/destructor
24         atmosphericLightQueue(int length);
25
26         void     insertValue(int dark, int r, int g, int b);
27         int              getAtmosphericLightR();
28         int              getAtmosphericLightG();
29         int              getAtmosphericLightB();
30         void     calcAtmosphericLight();
31         void     setMaxLength(int length){m_maxLength = length;}; //must be
               greater than 1
32         void     printList();
33         void     printRGBList();
34         void     clearList();
35         int              getLength();
36         int              getRealLength();
37 };
38 #endif
```

C. Appendix - AtmosphericLightElement Class

C.1. AtmosphericLightElement.h

```
1  #pragma once
2  #ifndef ATMOSPHERICLIGHTELEMENT_H
3  #define ATMOSPHERICLIGHTELEMENT_H
4
5  class atmosphericLightElement
6  {
7  public:
8
9          int    darkchannel;
10         int    r_input, g_input, b_input;
11         float intensity_input;
12         atmosphericLightElement* next, prev;
13
14         //constructor/destructor
15         atmosphericLightElement(int dark, int r, int g, int b)
16         {
17                 this->darkchannel = dark;
18                 this->r_input = r;
19                 this->g_input = g;
20                 this->b_input = b;
21                 this->intensity_input = 0.299f*r + 0.587f*g + 0.114f*b;
22                 //this->intensity_input = r + g + b; //this should be ok
                       too
23                 this->next = this->prev = NULL;
24         }
25 };
26 #endif
```

D. Appendix - Brightness Correction Functions

D.1. Gamma Correction

D.1.1. InitGammaCorrection()

```
1  void CGLDialog::InitGammaCorrection()
2  {
3          int max_exp = 20;
4
5          for(int greyTone=0; greyTone<256; greyTone++)
6          {
7                  for(int exp=0; exp<max_exp; exp++)
8                          m_gammaValues[greyTone][exp] = 255.0*pow((double)
                               greyTone/255.0, (double)exp*0.05);
9          }
10         TRACE(_T("DONE Initialising GammaCorrection.\n"));
11 }
```

D.1.2. CorrectGamma(..)

```
1  //m_gamma is a global variable, integer, between 0 and 20
2  void CGLDialog::CorrectGamma(int size)
3  {
4          if(m_gamma == 20) //when exp == 1
5                  return;
6
7          for (int x=0; x < size; x+=3)
8          {
```

```
9                   m_dehazedImageData[x     ] = m_gammaValues[
                        m_dehazedImageData[x   ]][m_gamma];
10                  m_dehazedImageData[x + 1] = m_gammaValues[
                        m_dehazedImageData[x+1]][m_gamma];
11                  m_dehazedImageData[x + 2] = m_gammaValues[
                        m_dehazedImageData[x+2]][m_gamma];
12          }
13  }
```

D.2. Histogram Equalisation

D.2.1. HistorgramEqualisationGrey(..) Greyscale Input

```
1   void CGLDialog::HistorgramEqualisationGrey(double* buf, int size)
2   {
3
4           int numberOfPixels = size;
5
6           int cdf[256];
7
8           for(int i=0; i<256; i++)
9           {
10                  cdf[i] = 0.0f;
11          }
12
13          //count the current frequencies
14          for(int i=0; i<size; i++)
15          {
16                  cdf[(int)(255.0*buf[i])]++; //count the frequencies
17          }
18
19          //calc cdf(cumulative distribution function) and store cdf in
                luminanceFrequency array
20          for(int i=1; i<256; i++)
21          {
22                  cdf[i] += cdf[i-1];
23          }
24
25          //remap the luminances
26          for(int i=0; i<size; i++)
```

```
27          {
28                  buf[i] = (((double)cdf[(int)(255.0*buf[i])] - cdf[0])/(
                        numberOfPixels - cdf[0]));
29          }
30  }
```

D.2.2. HistorgramEqualisationGrey(..) Colour Input

```
1   void CGLDialog::HistorgramEqualisationGrey(int size)
2   {
3
4           int numberOfPixels = m_camBuffer->getHeight() * m_camBuffer->
                getWidth();
5
6           int cdf[256];
7
8           for(int i=0; i<256; i++)
9           {
10                  cdf[i] = 0.0f;
11          }
12
13          //count the current frequencies
14          for(int i=0; i<size; i+=3)
15          {
16                  cdf[(m_dehazedImageData[i]+m_dehazedImageData[i+1]+
                        m_dehazedImageData[i+2])/3]++; //count the frequencies
17          }
18
19          //calc cdf(cumulative distribution function) and store cdf in
                luminanceFrequency array
20          for(int i=1; i<256; i++)
21          {
22                  cdf[i] += cdf[i-1];
23          }
24
25          //remap the luminances
26          for(int i=0; i<size; i+=3)
27          {
28                  m_dehazedImageData[i    ] = m_dehazedImageData[i + 1] =
29                          m_dehazedImageData[i + 2] = (((double)cdf[(
                                m_dehazedImageData[i]+m_dehazedImageData[i+1]+
                                m_dehazedImageData[i+2])/3] - cdf[0])/(
```

```
                                numberOfPixels - cdf[0])) * 255.0;
30          }
31  }
```

D.2.3. HistorgramEqualisationHSV(..)

```
1   void CGLDialog::HistorgramEqualisationHSV(int size)
2   {
3           double numberOfPixels = m_camBuffer->getHeight() * m_camBuffer->
                getWidth();
4
5           int cdf[256];
6
7           for(int i=0; i<256; i++)
8           {
9                   cdf[i] = 0;
10          }
11
12          //count the current frequencies
13          for(int i=0; i<size; i+=3)
14          {
15                  int r = m_dehazedImageData[i];
16                  int g = m_dehazedImageData[i+1];
17                  int b = m_dehazedImageData[i+2];
18
19                  //get max
20                  int max = r;
21                  if (g > max)
22                          max = g;
23                  if (b > max)
24                          max = b;
25
26                  cdf[max]++;
27          }
28
29          //calc cdf(cumulative distribution function) and store cdf in
                luminanceFrequency array
30          for(int i=1; i<256; i++)
31          {
32                  cdf[i] += cdf[i-1];
33          }
34
```

```
35          //remap the luminances
36          for(int i=0; i<size; i+=3)
37          {
38                  double r = m_dehazedImageData[i];
39                  double g = m_dehazedImageData[i+1];
40                  double b = m_dehazedImageData[i+2];
41
42                  //get min
43                  double min = r;
44                  if (g < min)
45                          min = g;
46                  if (b < min)
47                          min = b;
48
49                  //get max
50                  double max = r;
51                  if (g > max)
52                          max = g;
53                  if (b > max)
54                          max = b;
55
56                  double h = 0.0;
57
58                  if(max == min)
59                          h = 0.0;
60                  else if(max == r)
61                  {
62                          h = ((g - b) * 60.0 / (max - min));
63                          h =  (int)h % 360;
64                  }
65                  else if(max == g)
66                          h = ((b - r) * 60.0 / (max - min)) + 120.0;
67                  else if(max == b)
68                          h = ((r - g) * 60.0 / (max - min)) + 240.0;
69
70                  double v = max;
71                  double s = 0.0;
72                  if (max == 0.0)
73                          s = 0.0;
74                  else
75                          s = 1.0 - (min/max);
76
```

```
77                  double new_v = (((double)cdf[(int)v] - (double)cdf[0])/(
                        numberOfPixels - (double)cdf[0])) * 255.0;
78
79                  int hi = ((int)(h / 60.0)) % 6;
80                  double f = (h / 60.0) - floor(h /60.0);
81                  double p = new_v * (1.0 - s);
82                  double q = new_v * (1.0 - (f*s));
83                  double t = new_v * (1.0 - ((1.0 - f)*s));
84
85                  if(hi == 0)
86                  {
87                          m_dehazedImageData[i   ] = new_v;
88                          m_dehazedImageData[i+1] = t;
89                          m_dehazedImageData[i+2] = p;
90                  }
91                  if(hi == 1)
92                  {
93                          m_dehazedImageData[i   ] = q;
94                          m_dehazedImageData[i+1] = new_v;
95                          m_dehazedImageData[i+2] = p;
96                  }
97                  if(hi == 2)
98                  {
99                          m_dehazedImageData[i   ] = p;
100                         m_dehazedImageData[i+1] = new_v;
101                         m_dehazedImageData[i+2] = t;
102                 }
103                 if(hi == 3)
104                 {
105                         m_dehazedImageData[i   ] = p;
106                         m_dehazedImageData[i+1] = q;
107                         m_dehazedImageData[i+2] = new_v;
108                 }
109                 if(hi == 4)
110                 {
111                         m_dehazedImageData[i   ] = t;
112                         m_dehazedImageData[i+1] = p;
113                         m_dehazedImageData[i+2] = new_v;
114                 }
115                 if(hi == 5)
116                 {
117                         m_dehazedImageData[i   ] = new_v;
```

```
118                             m_dehazedImageData[i+1] = p;
119                             m_dehazedImageData[i+2] = q;
120                     }
121             }
122     }
```

D.2.4. HistorgramEqualisationRGB(..)

```
1   void CGLDialog::HistorgramEqualisationRGB(int size)
2   {
3
4           int numberOfPixels = m_camBuffer->getHeight() * m_camBuffer->
                getWidth();
5
6           int cdf_r[256];
7           int cdf_g[256];
8           int cdf_b[256];
9
10          for(int i=0; i<256; i++)
11          {
12                  cdf_r[i] = 0.0f;
13                  cdf_g[i] = 0.0f;
14                  cdf_b[i] = 0.0f;
15          }
16
17          //count the current frequencies
18          for(int i=0; i<size; i+=3)
19          {
20                  cdf_r[m_dehazedImageData[i  ]]++; //count the frequencies
                        for r
21                  cdf_g[m_dehazedImageData[i+1]]++; //count the frequencies
                        for g
22                  cdf_b[m_dehazedImageData[i+2]]++; //count the frequencies
                        for b
23          }
24
25          //calc cdf(cumulative distribution function) and store cdf in
                luminanceFrequency array
26          for(int i=1; i<256; i++)
27          {
28                  cdf_r[i] += cdf_r[i-1];
29                  cdf_g[i] += cdf_g[i-1];
```

```
30                  cdf_b[i] += cdf_b[i-1];
31          }
32
33          //remap the luminances
34          for(int i=0; i<size; i+=3)
35          {
36                  double r = (((double)cdf_r[m_dehazedImageData[i    ]] - (
                        double)cdf_r[0])/((double)numberOfPixels - (double)
                        cdf_r[0])) * 255.0;
37                  double g = (((double)cdf_g[m_dehazedImageData[i + 1]] - (
                        double)cdf_g[0])/((double)numberOfPixels - (double)
                        cdf_g[0])) * 255.0;
38                  double b = (((double)cdf_b[m_dehazedImageData[i + 2]] - (
                        double)cdf_b[0])/((double)numberOfPixels - (double)
                        cdf_b[0])) * 255.0;
39
40                  m_dehazedImageData[i    ] = r;
41                  m_dehazedImageData[i + 1] = g;
42                  m_dehazedImageData[i + 2] = b;
43          }
44  }
```

D.3. Histogram Splay

D.3.1. HistorgramSplayGrey(..)

```
1   void CGLDialog::HistorgramSplayGrey(double* buf, int size)
2   {
3
4           double min_luminance = 1.0;
5           double max_luminance = 0.0;
6
7           //get min and max value
8           for(int i=0; i<size; i++)
9           {
10                  double v = buf[i];
11
12                  if(v > max_luminance)
13                          max_luminance = v;
14                  if(v < min_luminance)
```

```
15                          min_luminance = v;
16          }
17
18          //remap the luminances
19          for(int i=0; i<size; i++)
20          {
21                  double v = buf[i];
22                  double new_v = (v - min_luminance)/(max_luminance -
                        min_luminance);
23                  buf[i] = new_v;
24          }
25  }
```

D.3.2. HistorgramSplayHSV(..)

```
1   void CGLDialog::HistorgramSplayHSV(BYTE* buf, int size)
2   {
3
4           double min_luminance = 255.0;
5           double max_luminance = 0.0;
6
7           //get min and max value
8           for(int i=0; i<size; i+=3)
9           {
10                  double v = buf[i];
11                  if(buf[i+1]>v)
12                          v = buf[i+1];
13                  if(buf[i+2]>v)
14                          v = buf[i+2];
15
16                  if(v > max_luminance)
17                          max_luminance = v;
18                  if(v < min_luminance)
19                          min_luminance = v;
20          }
21
22          //remap the luminances
23          for(int i=0; i<size; i+=3)
24          {
25                  double r = buf[i];
26                  double g = buf[i+1];
27                  double b = buf[i+2];
```

```
28
29                  //get min
30                  double min = r;
31                  if (g < min)
32                          min = g;
33                  if (b < min)
34                          min = b;
35
36                  //get max
37                  double max = r;
38                  if (g > max)
39                          max = g;
40                  if (b > max)
41                          max = b;
42
43                  double h = 0.0;
44
45                  if(max == min)
46                          h = 0.0;
47                  else if(max == r)
48                  {
49                          h = ((g - b) * 60 / (max - min));
50                          h = int(h) % 360;
51                  }
52                  else if(max == g)
53                          h = ((b - r) * 60.0 / (max - min)) + 120.0;
54                  else if(max == b)
55                          h = ((r - g) * 60.0 / (max - min)) + 240.0;
56
57                  int old_v = max;
58                  double s = 0.0;
59                  if (max == 0.0)
60                          s = 0.0;
61                  else
62                          s = 1.0f - (double)(min/max);
63
64                  double new_v = 255.0*((old_v - min_luminance)/(
                        max_luminance - min_luminance));
65
66                  int hi = ((int)(h / 60.0)) % 6;
67                  double f = (h / 60.0) - floor((double)h /60.0);
68                  double p = new_v * (1.0 - s);
```

```
69                  double q = new_v * (1.0 - (f*s));
70                  double t = new_v * (1.0 - ((1.0 - f)*s));
71
72                  if(hi == 0)
73                  {
74                          buf[i   ] = new_v;
75                          buf[i+1] = t;
76                          buf[i+2] = p;
77                  }
78                  if(hi == 1)
79                  {
80                          buf[i   ] = q;
81                          buf[i+1] = new_v;
82                          buf[i+2] = p;
83                  }
84                  if(hi == 2)
85                  {
86                          buf[i   ] = p;
87                          buf[i+1] = new_v;
88                          buf[i+2] = t;
89                  }
90                  if(hi == 3)
91                  {
92                          buf[i   ] = p;
93                          buf[i+1] = q;
94                          buf[i+2] = new_v;
95                  }
96                  if(hi == 4)
97                  {
98                          buf[i   ] = t;
99                          buf[i+1] = p;
100                         buf[i+2] = new_v;
101                 }
102                 if(hi == 5)
103                 {
104                         buf[i   ] = new_v;
105                         buf[i+1] = p;
106                         buf[i+2] = q;
107                 }
108         }
109 }
```

Bibliography

[Becker-Carus, 2004] Becker-Carus, C. (2004) *Allgemeine Psychologie* Elsevier GmbH, Spektrum Akademischer Verlag, München. 6, 10

[Blackwell, 1946] Blackwell, H. R. (1946) "Contrast thresholds of the human eye." *JOSA* 36: 624-643.

[Bouguer, 1760] Bouguer, P. (1760) *Traité d'otique sur la gradation de la lumière.* Paris, H.L. Guerin et L.F. Delatour. 2

[Brumberger et al., 1968] Brumberger, H. et al. (1968) "Light scattering," *Sci. Technol.*, November, pp. 34-60. 29

[Cadle, 1966] Cadle, R.D. (1966) *Particles in the Atmosphere and Space.* Van Nostrand Reinhold, New York.

[cambridgeincolour.com, 2011] cambridgeincolour.com 08-02-2011 21:13 PM `HTTP://WWW.CAMBRIDGEINCOLOUR.COM/TUTORIALS/UNSHARP-MASK.HTM`. 55

[Carr and Hartley, 2009] Carr, P. and Hartley, R. (2009) "Improved Single Image Dehazing Using Geometry," *2009 Digital Image Computing: Techniques and Applications,* dicta, pp.103-110. 63, 65, 69, 70, 71, 72, 75, 80

[Chavez, 1988] Chavez, P. S. (1988) "An improved dark-object subtraction technique for atmospheric scattering correction of multispectral data," Remote Sensing of Environment 24, 450-479. 52, 63, 67

[Clark and Whitby, 1967] Clark, W. M. and Whitby, K. T. (1967) "Concentration and

size distribution measurements of atmospheric aerosols and a test of the theory of self-preserving size distributions." *J. Atmos. Sci.*, 24, 677-687.

[Deirmendjian, 1963] Deirmendjian, D. (1963) "Scattering and polarization properties of polydisperseed suspensions with partial absorption." In *Electromagnetic Scattering*, M. Kerker, ed. Macmillan, New York.

[Dessens, 1946] Dessens, H. (1946) "Les noyaux de condensation de l'atmosphère." *CR* 223:915-917. 19, 26

[Dessens, 1947] Dessens, H. (1947) "Brume et noyaux de condensation." *Ann de Géophys.* 3: 68-86.

[Dickson et al., 1961] Dickson D. R. and Hales J. V. (1961) "Computation of Visual Range in Fog and Low Clouds." *Scientific Report for Armed Services Technical Information Agency Virginia.* 3. 47

[Duntley, 1948] Duntley, S. Q. (1948) "The reduction of apparent contrast by the atmosphere." *JOSA* 38:179-191.

[Eldridge, 1969] Eldridge, R. G. (1969) "Mist - The transition from haze to fog." *Bull. Am. Meteorol. Soc.*, 50, 422-426. 43

[Essentials of Physical Geography, 2006] Gabler,R. E., Petersen, J. F., Trapasso, L. M. (2006) *Essentials of Physical Geography* Essentials of Physical Geography Edition 8, Cengage Learning ISBN:0495011940. 15

[Fattal, 2008] Fattal, R. 2008 *Single Image Dehazing.* SIGGRAPH 2008 #0394 (online article: `HTTP://WWW.CS.HUJI.AC.IL/~RAANANF/PAPERS/DEFOG.PDF`). 3, 51, 52, 53, 62, 63, 66, 67, 77

[Fang et al., 2010] Fang, S., Zhan, J., Cao, Y., Rao, R. (2010) "Improved Single Image Dehazing Using Segmentation." Proceedings of 2010 IEEE 17th International Conference on Image Processing, September 26-29, 2010, Hong Kong. 105

[Fechner, 1859] Fechner, G. T. (1859) "Über ein psychologisches Grundgesetz." *Leipzig Abh.* 4: 457-532.

[Foitzik, 1938] Foitzik, L. (1938) "Über die Lichtdurchlässigkeit der stark getrübten Atmosphäre im sichtbaren Spektralbereich." *Wiss. Abh. Reichsamt. Wetterdienst.* Berlin, 4, No. 5. 43

[George, 1951] George, J.J. (1951) "Fog." *In Compendium of Meteorology* T.F. Malone, ed. AMS, Boston. 25

[Granath and Hulbert, 1929] Granath, L. P. and Hulbert, E. O. (1929) "The absorption of light by fog." *Phys. Rev.* 34:140-144.

[Green, 1932] Green, H. N. (1932) "The atmospheric transmission of coloured light." *R.A.E. Report E. and I.* 720.

[Gruner, 1919] Gruner, P. (1919) "Über die Gesetze der Beleuchtung der irdischen Atmosphäre durch das Sonnenlicht." *Beitr. Phys. freien Atm.* 8:120-155.

[Hautière and Aubert, 2005] Hautière, N. and Aubert, D. (2005) "Contrast Restoration of Foggy Images through use of an Onboard Camera," *Proceedings of the 8th International IEEE Conference of Intelligent Transportation Systems* Vienna, Austria, Spetember 13-16, 2005, pp. 1090-1095. 97, 115

[He et al., 2010a] He, K., Sun, J. and Tang, X. (2010) "Single Image Haze Removal Using Dark Channel Prior" *Pattern Analysis and Machine Intelligence, IEEE Transactions,* Issue: 99:1. 1, 54, 63, 66, 68, 69, 70, 75, 115

[He et al., 2010b] He, K., Sun, J. and Tang, X. (2010) "Supplemental Materials" `HTTP://PERSONAL.IE.CUHK.EDU.HK/~HKM007/CVPR09/`. 52, 53, 54

[Houghton, 1939] Houghton, J. G. (1939) "On the relation between visibility and the constitution of clouds and fog.", *J. Aer. Sci.* 6:408-411. 26

[Hulbert, 1941] Hulbert, E. O. (1941) "Optics at atmospheric haze" *J. Opt. Soc. Am.* 31: 467-476. 1, 2

[Huschke, 1959] Huschke, R. E. (1959) *Glossary of Meteorology* AMS, Boston. 13

[IPCC, 2000] IPCC(Intergovernmental Panel on Climate Change) (2000) "Aviation

and the Global Atmosphere", Chapter 3, http://www.grida.no/publications/other/ipcc_sr/?src=/climate/ipcc/aviation/032.htm. 18

[Joshi and Cohen, 2010] Joshi, N. and Cohen, M.F. (2010) "Seeing Mt. Rainier: Lucky imaging for multi-image denoising, sharpening, and haze removal," *Computational Photography (ICCP), 2010 IEEE International Conference,* 10.1109/ICCPHOT.2010.5585096, pp. 1-8. 114

[Junge, 1960] Junge, C.E. (1960) "Aerosols." In *handbook of Geophysics.*, C.F. Campen et al., eds. Macmillan, New York. 22

[Kopf et al., 2008] Kopf, J., Neubert, B., Chen, B., Cohen, M., Cohen-Or, D., Deussen, O., Uyttendaele, M., and Lischinski, D. (2008) "Deep photo: Model-based photograph enhancement and viewing." SIGGRAPH Asia 2008, http://johanneskopf.de/publications/deep_photo/index.html. 53, 66, 67, 115

[Koschmieder, 1924] Koschmieder, H. (1924) "Theorie der horizontalen Sichtweite" *Beitr. Phys. freien Atm.* 12:33-53; 171-181.

[Knoll et al., 1946] Knoll, H. A., Tousey, R., and Hulbert, E. O. (1946) "Visual thresholds of steady point sources of light in fields of brightness from dark to daylight." *JOSA* 36:480-482.

[Knowledgerush.com, 2011] www.knowledgerush.com/kr/encyclopedia/Eye/ 16-01-2011 4:14 PM. http://www.knowledgerush.com/wiki_image/e/ed/Human_eye_cross-sectional_view_grayscale.png. 7

[Laserpointerforums.com, 2010] www.laserpointerforums.com 08-05-2010 02:08 PM. http://laserpointerforums.com/attachments/f45/28579d1281013737-nichia-ship-green-diode-laser-vision-response-curve1.gif. 9

[Levin et al., 2006] Levin, A., Lischinski, D., Weiss, Y. (2006) "A closed form solution to natural image matting." *CVPR,* 1:61-68. 79

[Lv et al., 2010] Lv, X., Chen, W., Shen, I. (2010) "Real-Time Dehazing for Image

and Video," pacific_graphics, pp.62-69, 2010 18th Pacific Conference on Computer Graphics and Applications. 74, 98

[Manson et al., 1961] Manson, J.E. et al. (1961) "The possible role of gas reactions in the formation of the stratospheric aerosol layer." In *Chemical Reactions in the Lower and Upper Atmosphere.* Interscience, New York.

[McCartney, 1976] McCartney, E. J. (1976) *Optics of the Atmosphere.* John Wiley & Sons, New York / London / Sydney / Toronto. 4, 5, 10, 12, 15, 16, 17, 20, 21, 31, 35, 42

[Middleton, 1952] Middleton, W.E.K. (1952) *Vision through the Atmosphere.* University of Toronto Press, Toronto. 1, 4, 11, 15, 20, 30, 45

[Mie, 1908] Mie, G. (1908) "Beiträge zur Optik trüber Medien, speziell kolloidaler Metallösungen." *Ann. der Phys.* 25:377-445.

[Minnaert, 1992] Minnaert, M. G. J. (1992) *Licht und Farbe in der Natur.* Birkhäuser, Basel.

[Namer and Schechner, 2005] Namer, E. and Schechner, Y. Y. (2005) "Advanced Visibility Improvement Based on Polarizazion Filtered Images." Dept. Electrical. Eng., Technion - Israel Institute of Technology, Haifa, ISRAEL. 1, 54, 58, 60, 61, 73

[Narasimhan and Nayar, 2002] Narasimhan, S. G. and Nayar, S. K "Vision and the Atmosphere," International Journal of Computer Vision, v.48 n.3, p.233-254, July-August 2002. 62

[Narasimhan and Nayar, 2003] Narasimhan, S. G. and Nayar, S. K (2003) "Interactive deweathering of an image using physical models," *Workshop on Color and Photometric Methods in Computer Vision.* 73

[Nayar and Narasimhan, 1999] Nayar, S. K. and Narasimhan, S. G. (1999) "Vision in Bad Weather," The Proceedings of the Seventh IEEE International Conference on Computer Vision, September, 1999, pp. 820-827. 62, 63

[Neiburger and Wurtele, 1949] Neiburger, M. and Wurtele, M.G. (1949) "On the nature

and size of particles in haze, fog, and stratus of the Los Angeles region," *Chem. Rev.*, 44, 321-335. 25

[Online Mie Calculator, 2007] Scott, P. (2007) HTTP://OMLC.OGI.EDU/CALC/MIE_CALC.HTML. 41

[Petterson, 1960] Petterson, H. (1960) "Cosmic spherules and meteoristic dust", *Sci. Am.*, February, pp. 123-132.

[Rayleigh, 1871] Rayleigh (Lord) (1871) "On the scattering of light by small particles." *Phil. Mag.* 41:447-454. (*Collected Works*, 1:104-110.). 2

[Rayleigh, 1899] Rayleigh (Lord) (1899) "On the transmission of light through an atmosphere containing small particles in suspension, etc." *Phil. Mag.* 47:375-384. (*Collected Works*, 4:397-405.).

[Rossum and Nieuwenhuizen, 1999] Rossum, M. V., and Nieuwenhuizen, T. (1999) Multiple scattering of classical waves: microscopy, mesoscopy and diffusion, vol. 71, 313-371. 62

[de Saussure, 1789] de Saussure, H. B. (1789) "Descrpition d'un diaphanomètre ou d'un appareil propre à mesurer la transparence de l'air." *Mem. de l'Acad. de Turin* 4:425-440. 2

[Shwartz et al., 2006] Shwartz, S., Namer, E. and Schechner, Y.Y. (2006) "Blind Haze Separation," *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference* 10.1109/CVPR.2006.71, pp. 1984 - 1991. 63

[Schechner and Averbuch, 2007] Schechner, Y.Y. and Averbuch, Y. (2007) "Regularized Image Recovery in Scattering Media," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 29, no. 9, pp. 1655-1660, doi:10.1109/TPAMI.2007.1141. 113

[Schechner and Karpel, 2004] Schechner, Y. Y. and Karpel, N. (2004) "Recovering Scenes by Polarization Analysis," Proc. MTS/IEEE Oceans'04, Vol. 3, pp. 1255-1261. 60

[Schechner and Karpel, 2004b] Schechner, Y. Y. and Karpel, N. (2004) "Clear Underwater Vision" Proc. Computer Vision & Pattern Recognition, Vol. I, pp. 536-543. 115

[Schechner, et. al., 2001] Schechner, Y. Y., Narasimhan, S. G., and Nayar, S. K. (2001) "Instant dehazing of images using polarization," 325-332. 52

[Schechner, et. al., 2003] Schechner, Y. Y., Narasimhan, S. G., and Nayar, S. K. (2003) "Polarization-based vision through haze," *Optical Society of America,* 20. January 2003, Vol. 42, No. 3, Applied Optics. 60, 61

[Strong, 1958] Strong, J. (1958) *Concepts of Classical Optics*, W. H. Freman, San Fransisco. 32

[Tan, 2008] Tan, R. T. (2008) "Visibility in bad weather from a single image," *Proceedings of IEEE CVPR.* 53, 64, 65, 67, 68, 70, 77

[wikipedia.org, 2011a] en.wikipedia.org 31-03-2011 23:14 PM http://en.wikipedia.org/wiki/File:Rayleigh_sunlight_scattering.png. 33

[wikipedia.org, 2011b] en.wikipedia.org 07-02-2011 23:40 PM http://en.wikipedia.org/wiki/File:Gamma06_600.png. 56

[wikipedia.org, 2011c] en.wikipedia.org 07-02-2011 23:52 PM http://en.wikipedia.org/wiki/File:Histogrammspreizung.png. 57

[Wright, 1940] Wright, H. L. (1940) "Atmospheric opacity at Valentia." *Quart. J. Roy. Meteorol. Soc.*, 66, 66-77.