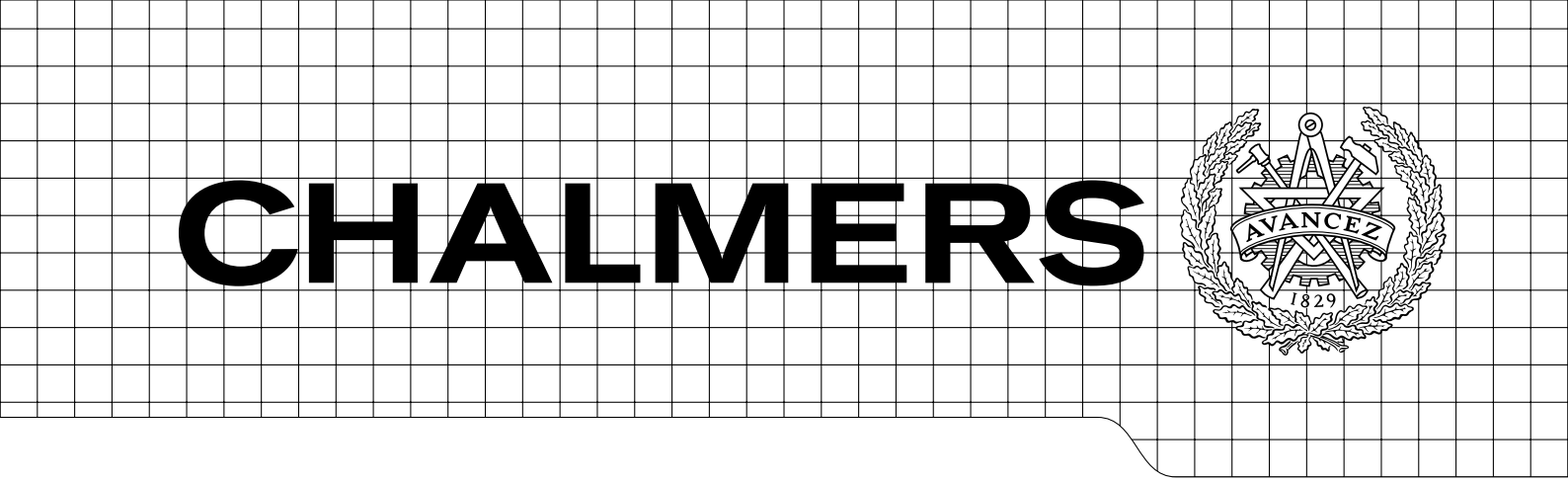


Contrast Enhancement, Denoising and Fusion in Dark Video

For Applications in Automobile Safety

Master's thesis in Signal Processing

NILS JUNGENFELT
TOBIAS RASKI

Department of Signals and Systems
Division of Signal Processing and Biomedical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2012
Master's thesis EX052/2012

MASTER'S THESIS IN SIGNAL PROCESSING

Contrast Enhancement, Denoising and Fusion in Dark Video

For Applications in Automobile Safety

NILS JUNGENFELT
TOBIAS RASKI

Department of Signals and Systems
Division of Signal Processing and Biomedical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY

Gothenburg, Sweden 2012

Contrast Enhancement, Denoising and Fusion in Dark Video
For Applications in Automobile Safety
NILS JUNGENFELT
TOBIAS RASKI

Master's thesis EX052/2012
ISSN 1652-8557
Department of Signals and Systems
Division of Signal Processing and Biomedical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone: +46 (0)31-772 1000

Chalmers Reproservice
Gothenburg, Sweden 2012

Contrast Enhancement, Denoising and Fusion in Dark Video
For Applications in Automobile Safety
Master's thesis in Signal Processing
NILS JUNGENFELT
TOBIAS RASKI
Department of Signals and Systems
Division of Signal Processing and Biomedical Engineering
Chalmers University of Technology

Abstract

A five step algorithm for automatic enhancement of night videos captured by a vehicle mounted camera is presented. The camera has a flash equipment which enables it to capture two almost simultaneous feeds of different exposure levels. As part of the algorithm the two feeds are individually enhanced through the means of tone mapping and noise removal, before being fused together. By filtering both in the spatial and the temporal domain, the noise removal step makes use of the fact that two consecutive frames are expected to show high correlation. The fusion method we present merges the best parts of both video feeds while getting rid of problems such as under- or overexposure that might be present in the individual feeds.

The target application is automobile safety and therefore a crucial part of the algorithm proposed here deals with enhancing contrast in non-irradiated regions of the video feed. This enables the driver to detect (and react to) potential safety threats as early as possible. The final step of the procedure aims to redistribute the local average intensity level evenly across the video frame. This removes disturbing edge effects originating from (low beam) head lamps and enables the driver to focus on what is important.

Keywords: Video enhancement, Video fusion, Multiple exposure, Filtering, Denoising, Tone mapping, Automotive night vision

Acknowledgements

We would like to thank *Fraunhofer Chalmers Centre for Industrial Mathematics* (FCC) for the opportunity to carry out this diploma work at the Department of Systems and Data Analysis. Special thanks to our main supervisors Johan Karlsson and Mats Kvarnström who have provided us with ideas and feedback on which the success of this thesis project heavily relies.

We also thank our examiner at Chalmers, Tomas McKelvey at the Department of Signals and Systems, who has worked alongside Johan and Mats as a supervisor and provided us with greatly appreciated input.

Finally, we would like to thank *Volvo Car Corporation* (VCC) who are the drivers behind a larger project within which ours is a small part. VCC have provided us with the data used throughout the thesis and the project manager Konstantin Lindström has given us feedback.

Contents

1 Introduction

An innovation in the field of car safety is the vehicle mounted camera capturing a video feed that can be visualised to the driver. One example of this are cameras mounted in such a way that they are able to capture blind spots on a truck (e.g. [GTC02]). This report will focus on another situation where a specialised vision system can be used, namely during night time driving.

Visual conditions are clearly significant from a car safety perspective, but it is difficult to assess to what extent. One indication of it is given in the fact in the average EU-27, the fatality rate per 100 injury accidents is at least 1.3 times higher at night than during the day [Mol+08]. It is reasonable to assume that this is to a large extent due to the poorer conditions of visibility when compared to daytime driving.

The visual condition problem can be addressed by using a vehicle mounted camera connected to a specialised vision system that can aid the driver. However, to be able to present a video of an improved visual quality, several steps of video processing are usually necessary. An observation of a scene and a digital video of the same scene are usually strikingly different. For example, the high dynamic range needed to observe details in deep shadows as well as in direct sunlight is typically impossible to capture and depict on an ordinary display without some kind of processing of the video.

If one wants to present noisy videos or high-dynamic range images on an ordinary low dynamic range display, a tone mapping procedure is needed. Unprocessed and displayed on a common display with low dynamic range without the use of any kind of tone mapping, the visual quality becomes very low, making it difficult for a person to perceive the observed scene and to suitably react to possible safety threats. Tone mapping aims for adjusting the overall contrast between different luminance levels in the image while at the same time keeping (and in some cases magnifying) the contrast locally, i.e. for the details. To avoid noise amplification when tone mapping some kind of noise supression procedure is also often needed.

For video processing the algorithms typically include time integration to avoid frame-to-frame flicker, and to achieve better enhancement of details and reduction of noise.

This work is part of the innovation and research project called *Visual Quality Measures*, financed by Vinnova through the FFI program (2009-00071). Members in the project are VCC, FCC, Epsilon and Chalmers.

1.1 Purpose

In this report we study video signals captured at night by a vehicle mounted camera. The video signals considered are degraded by noise and subject to poor or non-uniform illumination. Unprocessed the visual quality of such a video is typically very low, making it difficult for a person to perceive the observed scene. To make the videos more visually suitable a tone mapping and noise supression procedure is needed.

Further, in the application at hand we work in a setting where we have two near-parallell video streams captured at different exposure levels. More specifically, one of the streams have been captured with a flash and the other without.

The flash video will typically be better illuminated in some parts of the observed scene in

comparison to the non-flash video but might suffer from over-exposures in other parts. A key part of the problem is to figure out how to fuse these two video streams in such a way that the best parts of both videos is preserved.

To solve all of these problems we will within this report present an algorithm, that when fed with two differently exposed video streams produces a video stream of higher visual quality, through the means of tone mapping, noise removal and fusion.

1.2 Limitations

Our work covers nothing beyond the realm of video processing, i.e we do not explore video analysis methods such as object detection etc. Neither do we explore how to present video data to potential car drivers.

In this report we will look at a particular video dataset which we have been using to design our algorithm, but we will not be terribly concerned with the equipment used to capture it. This is due to the fact that the actual equipment has not been available to us during our work.

As a final product, our work should ideally be implemented on custom designed hardware, but this also lies outside of our scope. As an extension of this we have not been very concerned with achieving a realtime implementation and the work presented in this report should largely be considered a proof-of-principle.

1.3 The outline of this report

This report is divided in five different chapters, including this introduction. In the next chapter we will start by briefly presenting some basic theory concerning digital image and video processing. The report is written with readers unfamiliar to this field of study in mind and contains only the basics needed to understand the rest of this report. The presentation is in no fashion or form complete, instead we refer the interested reader to [Bov05].

The third chapter covers the problem we have been working with together with the algorithm we have devised for solving it. The algorithm is the major result of the thesis project that lies behind this report and is presented in such a way that a reader should be able produce his or hers own implementation of it. The chapter also covers our implementation of the algorithm. At the end of the chapter we will discuss our work in comparison to a few other published approaches.

In the fourth chapter we take a look at what results we can produce using our approach and take a look at improvements compared to the raw input data. We also deconstruct the algorithm we present through a series of test cases in an effort to make the reader better understand why the different steps in our solution are sound ones.

Lastly, in the final chapter we give a reflection of our own work and offer opinions on how we think the things we have done can be extended in the future.

2 Theory

The aim of image/video processing is often to improve the visual quality, typically for a human observer and throughout this report some concepts from this field will arise time and time again and we need to familiarise ourselves with them. In this chapter we will start with the basics and describe the fundamentals of images and videos in a digital form, we will cover the basics of contrast and illumination enhancement and, after that, what we mean by *digital noise* and a few different ways of getting rid of it.

2.1 Basics of digital images and videos

In this report a digital image I (sometimes denoted J) is a two dimensional matrix of pixels, with some height n and some width m. We also adopt the standard way of thinking of I as a function $M \to V$, with $M = \{(i, j) : i \leq m, j \leq n\}$ and V being a set of pixel values, which means that by $I(i, j)$ we will denote the value of the pixel in row i and column j of image I.

Throughout this report we will only deal with greyscale images which means that all pixels have a single (but variable) value: the pixel's *intensity* or *grey level*. The pixels can take on intensity values within some predefined range V, e.g $V = [0, 255]$ in an 8-bit representation, where the smallest value in the range corresponds to black and the largest to white.

Extending this notion to videos, we consider a video $I = \{J_t\}$ to be a collection of images, or *frames*, i.e. a 3D matrix. Correspondingly we let $I(i, j, t)$ point to the pixel in row i and column j in frame number t of the video I. For brievity we will often when dealing with a video use the notation I_t when talking about the t:th frame of video I.

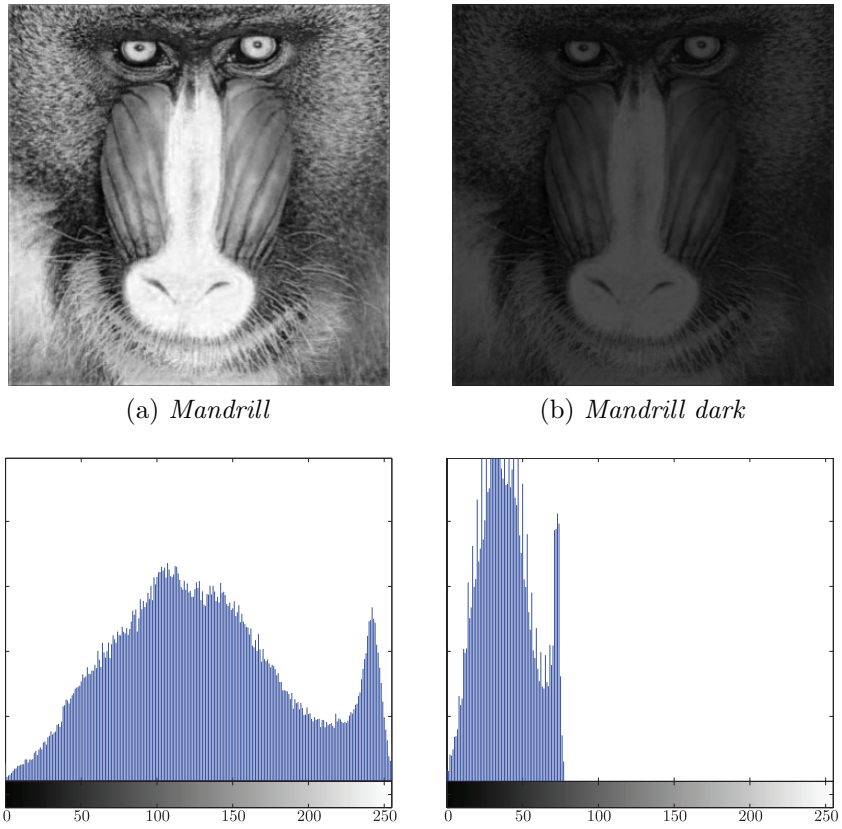
2.1.1 Image histogram

A basic tool for understanding and analysing the grey-levels in a greyscale image is the *image histogram*.

The histogram H_I of an image I is a graph that shows the *frequency of occurrence* of each grey level in I. More precisely, given an image I of size $m \times n$ and a grey level range of k different levels, H_I is a function $\{0, \ldots, k-1\} \to \{0, \ldots, mn\}$. This means that if a given grey level $q \in \{0, \ldots, k-1\}$ appears exactly j times in I we will have $H_I(q) = j$, where j can range from 0 to mn (the first meaning the grey level does not appear into the image at all, the latter that the image pixels all are constantly valued at q).

By just glancing at the histogram of a greyscale image one can often judge the tonal distribution of the same image. In Figure 2.1 we see two versions of an image, one much darker than the other. That fact is also apparent from the two image histograms where the left one is well distributed over all grey levels whereas the right one is skewed and heavy to the left making the image dark and lower in contrast.

Enhancement of images such as in Figure 2.1b with this kind of disadvantaged is an important part of this report and the basics of doing so will be covered in the next section.



(a) *Mandrill* (b) *Mandrill dark*

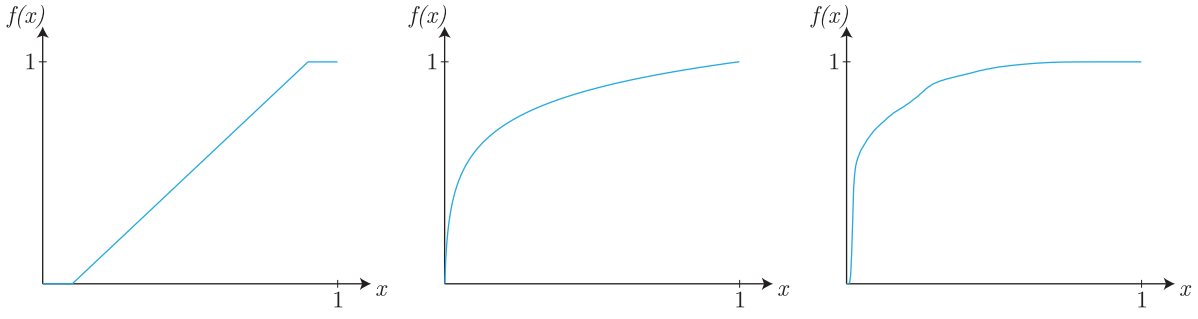
(c) *Mandrill histogram* (d) *Mandrill dark histogram*

Figure 2.1: *Image histogram*

2.2 Contrast enhancement

As seen in Figure 2.1b an image with an unwanted distribution of its pixel values often come with some problems. In this report we will work with greyscale images that typically has very dark or very bright regions from where it is difficult for the human eye to derive information due to the low contrast.

The approach to solve this problem is by transforming the pixel values using some transformation function [RC12]. Examples of such functions are shown in Figure 2.2, each function



(a) *Global contrast stretching* (b) *Logarithmic tone mapping* (c) *Histogram equalisation*

Figure 2.2: *Examples of transformation functions for contrast enhancement.*

derived from the three different enhancement methods that will be described in the next three sections.

2.2.1 Global contrast stretching

The simplest form of transformation function is linear, mapping pixels to new values such that typically 1% of them are saturated at high and low intensities. Assume, for example, that 98% of the pixels in an image lie within the intensity range $[0.1, 0.9]$, then a transformation function may look like Figure 2.2a.

2.2.2 Logarithmic tone mapping

In an image with large very dark regions (e.g. captured at night), one typically wants to increase contrast at the low intensity range only. A transformation that enhances contrast in the dark intensity range at the expense of contrast in the bright region is necessary. Hence, a strictly concave transformation function is the way to go here.

(a) *Original image* (b) *Tone mapped*

Figure 2.3: *An example of the effect of logarithmic tone mapping, using* $b = 2.5$ *in equation (2.1).*

The simplest choices for a strictly concave function is a piecewise linear one with decreasing inclination, or $f(x) = x^\alpha$ with $0 < \alpha < 1$. However, in image processing it is common to use a logarithmic function, e.g. as proposed by [Dra+03] and discussed in [Xu+10], a transformation function $f : [0, 1] \mapsto [0, 1]$ may have the form

$$f(x) = \frac{\ln(10)}{\ln(256)} \cdot \frac{\ln(255x + 1)}{\ln\left(5x^{\frac{\ln(b)}{\ln(0.5)}} + 5\right)} \tag{2.1}$$

for some design parameter $b \in [0.6, 4]$. The function is plotted in Figure 2.2b with $b = 1$, and Figure 2.3 shows a result from a tone mapping with $b = 2.5$.

2.2.3 Histogram equalisation

A more sophisticated way of enhancing the contrast of an image aims to redistribute the histogram uniformly over the interval. This is done by letting the transformation function be proportional to the corresponding cumulative distribution function (CDF) of the histogram, see Figure 2.2c.

An improvement of this approach is to transform each pixel with a locally derived transformation function based on the neighbourhood of the considered pixel. This method, known as adaptive histogram equalisation (AHE), is computationally expensive. To speed up such an algorithm, the image is partitioned into rectangular tiles and a transformation function is computed for each tile. The functions derived are appropriate for the center pixel in each tile, pixels in between are assigned interpolated values.

A magnification of an intensity range naturally increases the noise level in this range, signal to noise ratio being constant. If a tile is plain or has large plain regions, i.e. the histogram has one or more very high peaks, then the transformation function is very steep at those intensity levels and noise may be over-amplified. To solve this issue, contrast limited adaptive histogram equalisation (CLAHE) was proposed by [Zui94]. The idea is to clip the histogram at a predefined value, typically $0.01mn$ (which corresponds to 1% of pixels having exactly the same intensity), before computing the CDF. It is considered advantageous not to discard parts of the histogram that exceed the clip limit but to redistribute them equally among all histogram bins.

2.3 Image and video denoising

Raw data from a digital camera is typically degraded by *noise*, especially if captured under poor illumination conditions. Noise can arise due to many different reasons and can be roughly divided into *fixed pattern noise* and *random noise*. Fixed pattern noise is connected to properties of the camera equipment in use and is relatively easy to get rid of due to the fact that it is fixed and known. The real problem is random noise and that is consequently what we need to concern ourselves with.

A common type of random noise is *photon shot noise* which is a product of the statistical variations in the number of incoming photons at every detector, i.e. pixel, of the camera. This is typically adjusted for by slowing down the shutter speed [BM05], which for obvious reasons is a troublesome idea for applications where one needs crisp and non-blurry images.

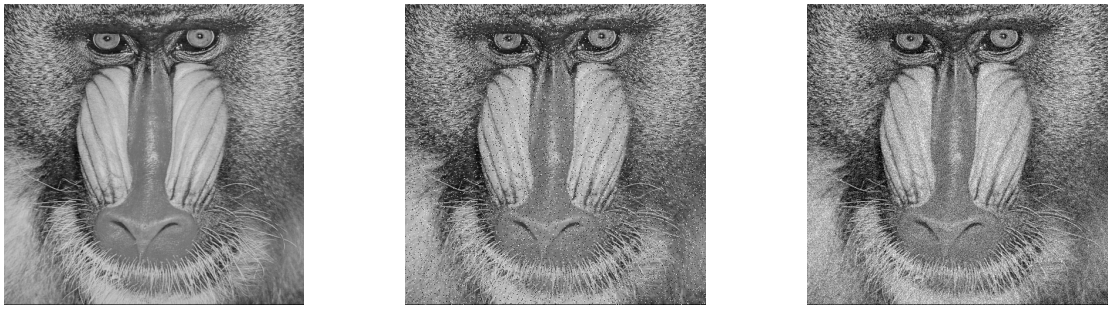
Random noise a from natural source (e.g shot noise) is often approximated with additive white Gaussian noise (AWGN) with zero mean, i.e.

$$I(i,j) = I^{\text{true}}(i,j) + Z(i,j) \tag{2.2}$$

where I is an image, I^{true} is the true signal, and $Z \sim \mathcal{N}(0, \sigma^2)$ is the noise.

Another common form of noise is the *salt and pepper noise* which presents itself as randomly occurring white or black pixels. Figure 2.4 shows an example of both Gaussian and salt and pepper noise.

Denoising, sometimes also called *filtering*, is a process whereby noise and other potential irregularities are attenuated from a digital image. The following subsections will deal with a few different approaches to denoising. First we will look at classical approaches with linear and nonlinear filters in Section 2.3.1 and after that at wavelet denoising and denoising using



(a) *Original greyscale image* (b) *Salt-and-pepper noise* (c) *Gaussian zero mean noise with varianece 0.01*

Figure 2.4: *Examples of noise.*

the non-locality idea in Section 2.3.2. In Section 2.3.3 we will take a look at how to extend denoising to the temporal dimension present in digital videos.

2.3.1 Simple linear and nonlinear filters

A filter F being *linear* simply means that for any two images I_1 and I_2, and any two constants a and b, F is subject to the following constraint:

$$aF(I_1) + bF(I_2) = F(aI_1 + bI_2) \tag{2.3}$$

Average and Gaussian filters

Consider a geometric rule B that defines how to select intensities from pixels that are in the the vicinity of a given pixel. B is a window since it will typically be the rule that given an image I and a pixel coordinate $\boldsymbol{n} = (i, j)$ it collects the intensities of the pixels that are in a square-shaped neighbourhood (of some given size) with its center in $\boldsymbol{n}$.

Now, let the moving average filter $\mathcal{A}$ be defined by:

$$I^{\text{avg}}(\boldsymbol{n}) = \mathcal{A}[B(I, \boldsymbol{n})] \tag{2.4}$$

where $\mathcal{A}$ computes the average for each local neighbourhood in I (i.e. for each coordinate (i, j)), producing a smoothing effect, the strength of which depends on the size of B.

Since taking the average is a linear operation we also have that

$$\mathcal{A}[B(I, \boldsymbol{n})] = \mathcal{A}[B(I^{\text{true}}, \boldsymbol{n})] + \mathcal{A}[B(Z, \boldsymbol{n})] \tag{2.5}$$

in a case where equation (2.2) holds, i.e. I^{true} is degraded by some noise Z. The standard assumption is that Z is of zero mean, which implies that when the size of B increases to infinity the last term in the above equation tends to zero (see [Bov05] for an explanation of this). Consequently the moving average filter has the desired property of reducing zero mean noise towards zero, but unfortunately it also has some flaws. We would like it to be that the case that at each $\boldsymbol{n}$ we have $\mathcal{A}[B(I^{\text{true}}, \boldsymbol{n})] \approx I^{\text{true}}(\boldsymbol{n})$. This is not the case since $\mathcal{A}$ will effect the original image information, especially if the size of B is large. For example, if I^{true} is a

scene with sharp edges $\mathcal{A}$ will do a poor job of not reducing that information. However, edge preservation is an important and desirable property of a filter.

The Gaussian filter is just a simple extension of the average filter. Above, the window B selects intensities from pixels in a neighbourhood with *equal* weights, i.e. every pixel in a neighbourhood contribute equally to the average. This is not necessary and we can let B select pixels with any weighting we want. If we let B select the center pixel of the neighbourhood with the highest weight, and the other pixels with weights that fall of in Gaussian manner from the center towards the edges of the neighbourhood, B becomes a Gaussian filter. More precisely, at a point $\boldsymbol{n} = (i, j)$ weights are selected to be proportional to

$$g(\boldsymbol{x}, \boldsymbol{n}) = e^{-\frac{d(\boldsymbol{x},\boldsymbol{n})^2}{2\sigma^2}} \tag{2.6}$$

where $d(\boldsymbol{x}, \boldsymbol{n}) = \|\boldsymbol{x} - \boldsymbol{n}\|$ is the Euclidean distance between the points $\boldsymbol{x}$ and $\boldsymbol{n}$, and σ is the standard deviation indicating the geometric spread of B. For computational efficiency, one may want to set a finite window size for B. Let $\boldsymbol{x} \in \Omega(\boldsymbol{n})$ denote the fact that $\boldsymbol{x}$ and $\boldsymbol{n}$ are neighbouring pixels according to the window size of B. The filtered image J is now given by

$$J(\boldsymbol{n}) = k^{-1}(\boldsymbol{n}) \sum_{\boldsymbol{x}\in\Omega(\boldsymbol{n})} g(\boldsymbol{x}, \boldsymbol{n}) \cdot I(\boldsymbol{n}) \tag{2.7}$$

where

$$k(\boldsymbol{n}) = \sum_{\boldsymbol{x}\in\Omega(\boldsymbol{n})} g(\boldsymbol{x}, \boldsymbol{n}). \tag{2.8}$$

Median filters

Another thing one can do is to exchange the $\mathcal{A}$ operation in the average filter for the median operation. One might guess that this should yield a similar result, but that is not necessarily the case. First of all the median operation is not linear and, more importantly, it is known to be quite effective at removing salt and peper noise without distorting edges [Bov05]. The median filter is also generally known to be better than any linear filter at removing medium level Gaussian noise (whilst preserving edges). For high levels of noise it is debatable if this is the case or not [Bov05], and in either case it is not particularly effective when compared to more advanced methods.

2.3.2 Other denoising techniques

The techniques presented above are ubiquitous in the image and video processing field. Also, they are simple and therefore fast and appropriate in hardware implementations. Unfortunately, the quality of the end results is substandard when compared to more advanced methods of denoising. Their main problem is that they are unable to remove heavy noise while still preserving edge detail.

There are numerous more or less advanced techniques that aim to solve this problem and in the next two sections we will take a look at two general methods, while at the same time arriving at the main method we have opted to use for our purposes.

Wavelet shrinkage

The method of wavelet shrinkage for signal denoising was originally developed by Donoho and Johnstone and it has since then been extensively explored, examples of which can be seen in [Por+03] and [SA96]. The method uses discrete wavelet transformation (DWT), which is advantageous over the more well known Fourier transform in that it holds both frequency and location information. The motivation for denoising in this domain is that signal and noise may be separated; the DWT compacts the energy of the true signal into a small number of DWT coefficients with large amplitudes, and spreads the energy of the noise over a large number of DWT coefficients having small amplitudes.

The key idea is to use a pointwise thresholding operator, Υ_λ, that sets DWT coefficients smaller than a threshold λ to zero while leaving large coefficients unchanged. The estimate of a true image is

$$\widehat{I^{\text{true}}} = \mathcal{W}^{-1}\Upsilon_\lambda \mathcal{W} I \tag{2.9}$$

where the operators $\mathcal{W}$ and $\mathcal{W}^{-1}$ are the forward and inverse discrete wavelet transforms, respectively, and I is a noisy image.

There are two common ways of constructing Υ_λ. One is known as *hard thresholding* and the operator is, for a given function $J(\boldsymbol{n})$, defined as

$$(\Upsilon_\lambda J)(\boldsymbol{n}) = \begin{cases} J(\boldsymbol{n}), & \text{if } |J(\boldsymbol{n})| > \lambda, \\ 0, & \text{otherwise.} \end{cases} \tag{2.10}$$

The *soft thresholding* operator, on the other hand, is defined as

$$(\Upsilon_\lambda J)(\boldsymbol{n}) = \begin{cases} J(\boldsymbol{n}) - \lambda, & \text{if } J(\boldsymbol{n}) > \lambda, \\ J(\boldsymbol{n}) + \lambda, & \text{if } J(\boldsymbol{n}) < -\lambda, \\ 0, & \text{otherwise.} \end{cases} \tag{2.11}$$

Bilateral filtering

The bilateral filter, introduced by [TM98], is derived from Gaussian blur but has the much desired property of being edge-preserving. The key idea is that the weight for a neighbouring pixel is depending not only on the Euclidean metric but also on the radiometric difference, i.e. the difference in intensity for a grey scale image.

First, consider a *closeness function*

$$g_d(\boldsymbol{x}, \boldsymbol{n}) = e^{-\frac{d(\boldsymbol{x},\boldsymbol{n})^2}{2\sigma_d^2}} \tag{2.12}$$

where $d(\boldsymbol{x}, \boldsymbol{n})$ is the Euclidean distance between the points $\boldsymbol{x}$ and $\boldsymbol{n}$, and σ_d is the geometric spread of the filter.

Second, we need a *similarity function*. For an image I it is given by

$$g_r(\boldsymbol{x}, \boldsymbol{n}) = e^{-\frac{\delta(I(\boldsymbol{x}),I(\boldsymbol{n}))^2}{2\sigma_r^2}} \tag{2.13}$$

where $\delta(a_1, a_2)$ is a suitable measure of distance between the intensity values a_1 and a_2, and σ_r is the photometric spread of the filter.

Now, consider a window $\Omega(\boldsymbol{n})$ of some finite size, surrounding pixel $\boldsymbol{n} = (i, j)$. The resulting image J of a bilateral filtration is then

$$J(\boldsymbol{n}) = k^{-1}(\boldsymbol{n}) \sum_{\boldsymbol{x} \in \Omega(\boldsymbol{n})} g_d(\boldsymbol{x}, \boldsymbol{n}) \cdot g_r(\boldsymbol{x}, \boldsymbol{n}) \cdot I(\boldsymbol{x}) \tag{2.14}$$

where

$$k(\boldsymbol{n}) = \sum_{\boldsymbol{x} \in \Omega(\boldsymbol{n})} g_d(\boldsymbol{x}, \boldsymbol{n}) \cdot g_r(\boldsymbol{x}, \boldsymbol{n}). \tag{2.15}$$

Non-locality

In recent years a new class of filters using the idea of non-locality has emerged, the forerunner being the Non-Local Means (NLM) method presented in [BCM+05]. The non-locality idea says that we can gain information about a given area of a frame or image by looking at areas similar to it, similarity being defined by some kind of metric.

In classical 2D NLM, the similarity function compares *patches* surrounding the pixels under consideration and it is given by

$$g(\boldsymbol{x}, \boldsymbol{n}) = e^{-\frac{\delta_I(\boldsymbol{x}, \boldsymbol{n})}{h^2}} \tag{2.16}$$

where

$$\delta_I(\boldsymbol{x}, \boldsymbol{n}) = \sum_{\boldsymbol{y} \in \Omega_C(\boldsymbol{0})} G_\sigma(\boldsymbol{y}) \cdot |I(\boldsymbol{x} + \boldsymbol{y}) - I(\boldsymbol{n} + \boldsymbol{y})|^2. \tag{2.17}$$

G_σ is a 2D Gauss kernel of standard deviation σ and Ω_C is a 2D neighbourhood domain defining the comparison window (i.e. the size of the patches). For a search window Ω_S, we have, just like in previous examples, a filtered image J by taking

$$J(\boldsymbol{n}) = k^{-1}(\boldsymbol{n}) \sum_{\boldsymbol{x} \in \Omega_S(\boldsymbol{n})} g(\boldsymbol{x}, \boldsymbol{n}) \cdot I(\boldsymbol{n}) \tag{2.18}$$

where

$$k(\boldsymbol{n}) = \sum_{\boldsymbol{x} \in \Omega_S(\boldsymbol{n})} g(\boldsymbol{x}, \boldsymbol{n}). \tag{2.19}$$

NLM can easily be adopted to video filtering (3D NLM) by employing a temporal search window Ω_T and performing the search in the 3D space (Ω_S, Ω_T).

This new class of denoising algorithms employing the non-locality statistics can fairly be said to be at the leading edge of the image denoising field. In particular BM3D [Dab+07] is considered by many to be the current state-of-the-art in image denoising [LDW09].

The BM3D method relies on two basic techniques, grouping and collaborative filtering. The algorithm starts of by dividing an image I into a set I_R of *reference blocks*, labeled such that $\boldsymbol{x}_R$ is the upper-left coordinate of the reference block $Z_{\boldsymbol{x}_R}$. The reference blocks are allowed to be overlapping, it is only important that each pixel in I is contained in at least one reference blocks.

For each $Z_{\boldsymbol{x}_R}$, similar blocks are found by block matching (the non-locality idea) in a given search area. In contrast to NLM we only consider the N most similar blocks (N being a design parameter) and stack these together in a so called group (3D array) $\mathcal{S}_{\boldsymbol{x}_R}$. The group is then

sorted by decreasing measure of similarity to form $\mathbf{Z}_{\mathcal{S}_{\boldsymbol{x}_R}}$ and undergoes the following procedure:

$$\widehat{\mathbf{Y}}_{\mathcal{S}_{\boldsymbol{x}_R}} = \mathcal{T}_{3D}^{-1}\Upsilon_\lambda\mathcal{T}_{3D}\mathbf{Z}_{\mathcal{S}_{\boldsymbol{x}_R}} \tag{2.20}$$

where $\mathcal{T}_{3D}$ is a separable 3D transform and Υ_λ the hard thresholding function from equation (2.10). $\widehat{\mathbf{Y}}_{\mathcal{S}_{\boldsymbol{x}_R}}$ is now a group of blockwise estimates $\widehat{Y}^{\boldsymbol{x}_R}_{\boldsymbol{x'}\in\mathcal{S}_{\boldsymbol{x}_R}}$. For future reference it will be easier if we think of a block estimate $\widehat{Y}^{\boldsymbol{x}_R}_{\boldsymbol{x'}}$ as being a matrix of the same size as I, but zero valued everywhere except at the coordinates in the block defined by $\boldsymbol{x'}$.

Each estimated block also comes with a weight $\omega_{\boldsymbol{x}_R}$ which corresponds to how reliable the estimate is considered to be. For the specifics of how to calculate the weights the reader is referred to [Dab+07].

After procedure 2.20 has been performed for all reference blocks, a denoised estimate $J = \widehat{I^{\text{true}}}$ can be derived from I by aggregation of all blockwise estimates $\widehat{Y}^{\boldsymbol{x}_R}_{\boldsymbol{x'}}$:

$$J(\boldsymbol{n}) = k^{-1}(\boldsymbol{n})\sum_{\boldsymbol{x}_R}\sum_{\boldsymbol{x'}\in\mathcal{S}_{\boldsymbol{x}_R}}\omega_{\boldsymbol{x}_R}\widehat{Y}^{\boldsymbol{x}_R}_{\boldsymbol{x'}}(\boldsymbol{n}) \tag{2.21}$$

where

$$k(\boldsymbol{n}) = \sum_{\boldsymbol{x}_R}\sum_{\boldsymbol{x'}\in\mathcal{S}_{\boldsymbol{x}_R}}\omega_{\boldsymbol{x}_R}\chi_{\boldsymbol{x'}}(\boldsymbol{n}), \tag{2.22}$$

$\chi_{\boldsymbol{x'}}$ being the characteristic function of the square support of the block located at $\boldsymbol{x'}$, i.e. $\chi_{\boldsymbol{x'}}(\boldsymbol{y}) = 1$ if $\boldsymbol{y}$ is in the block and otherwise 0.

The scheme described above is shown in a more brief form in Figure 2.5. In the figure, *ebuff* illustrates an image buffer of the block estimations calculated so far, *wbuff* is a buffer of corresponding weights.

The BM3D method has been generally praised and during our testing we have come to the same conclusion regarding its performance. Therefore, when we, starting from the next chapter in this report, talk about denoising will be referring to BM3D-denoising, or more specifically to the video-denoising extension of it, namely VBM3D.

2.3.3 Video denoising

In video filtering, if one only denoises spatially (frame by frame) it may be the case that one ends up with frames that individually look relatively noise free but when viewed in sequence appear very noisy due to the change in noise pattern between frames. In video denoising it is therefore common to use a filter that also works in the temporal dimension. Temporal denoising is a way of removing noise between frames, i.e. pixels that from one frame to another changes irregularly. The idea is that pixels in a given frame should, when denoised, be affected by pixels in past and/or future frames.

All of the above denoising methods can fairly easily be extended to also work in the temporal dimension. For example the window B we mentioned when talking about average filters can be extended to a 3D box/kernel that collects pixels close to each other from a spatial as well as temporal perspective.

Spatio-temoral denoising can also work better than only spatial denoising since it is expected that there should be some correlation between two consecutive frames in a video. We are therefore able to gain information about the observed scene in a given frame by looking at

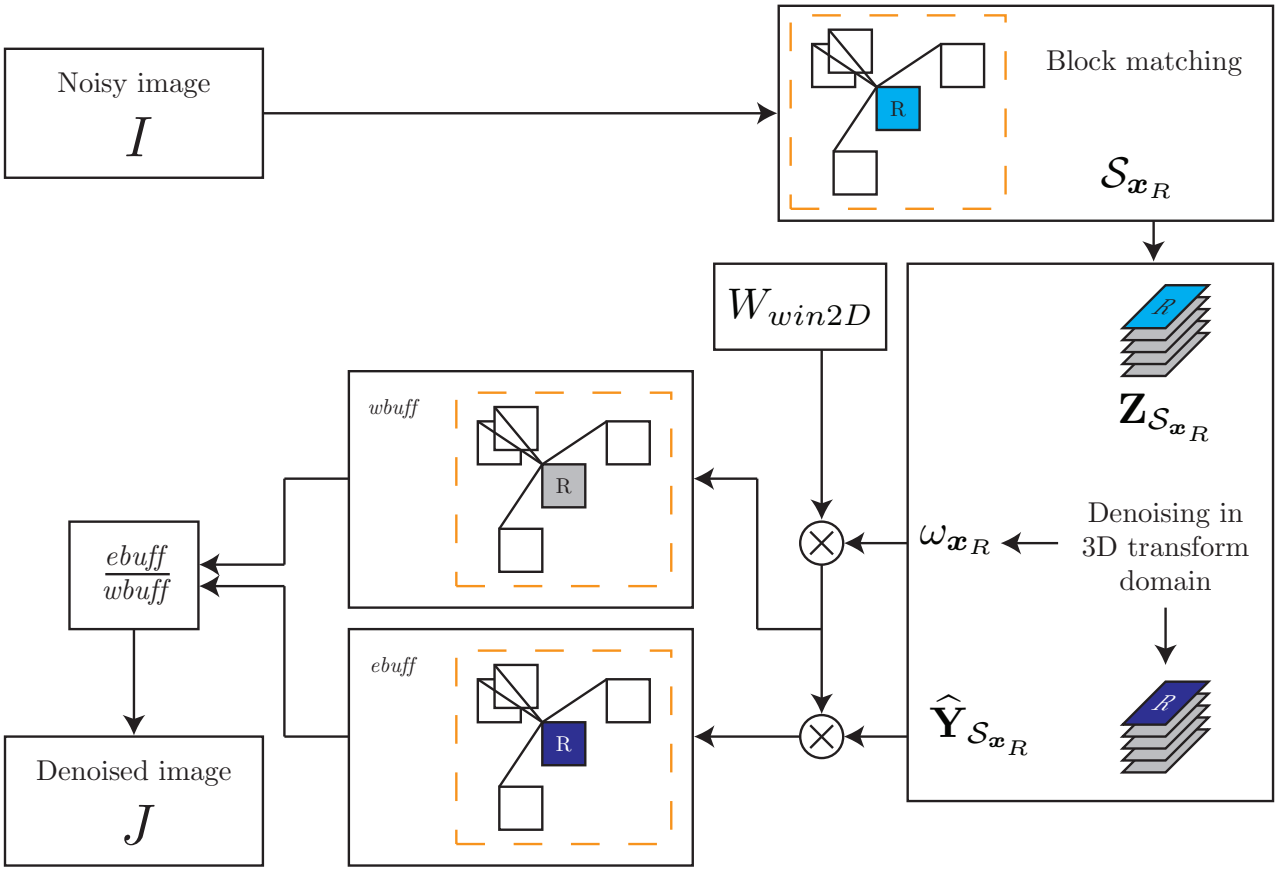


Figure 2.5: *Flowchart describing BM3D.*

the frames that came before and those that will come after. This of course depends on how well consecutive frames correlate, and in a very fast changing video one might not be so lucky. It might even be the case that when blending pixels from several frames together ghosting artifacts are created. To avoid this many spatio-temporal denoising methods has some kind of motion compensation that aims to predict the movement of pixels between frames.

The number of frames to take into account in this way so that the filter performs effectively depends highly on the observed scene. In a constant scene we expect to gain more from a high number where as in a scene with much change we will gain less. It should also be noted that a higher number of frames can lead to more costly computations.

3 Method

During our work we have studied noise degraded video signals captured under poor illumination conditions from a vehicle mounted camera driving at night and in this chapter and in the next few sections we will outline an algorithm for enhancing such video data. Before going in to the specifics of the algorithm lets take a look at the vehicle mounted camera of interest for us.

3.1 Camera and dataset

Commercial digital cameras typically uses CMOS or CCD image sensors that operate with an RGB colour model. A colour filter is a mosaic of small filters that is placed over the pixels of the image sensor in order to capture colour information. The Bayer filter is for example a common colour filter which gives RGB information and its mosaic structure is ordered such that one gets a filter pattern of 50 % green, 25 % red and 25 % blue.

In our application we have been interested in a camera that works with a non-colour model. The camera has 720×1280 pixels, divided among two kinds of pixel sensors, i.e. it has two channels. Only one of the channels will be of interest to us and it captures information that gives greyscale intensities. Dividing the pixels into 2×2 grids, three out of the four pixels corresponds to the channel we are interested in.

Just as in the case with a corresponding RGB setting we are left with a situation where our channel is smaller than the full pixel resolution. The standard solution to this is to use interpolation to complete (demosiac) each channel. We have however decided to simplify the setting a bit and drop the resolution to 360×640 by considering each of the three pixels of interest in a 2×2 grid to be three versions of the same scene and taking their average. This leaves us with single channeled grey scale data of size 360×640 and the reader can assume this to be the case through out the rest of this report. This decision was made in order to get data that is considerably easier to handle.

Furthermore, the camera comes with another novelty and can capture two almost parallel video feeds. One of the feeds is captured with the help of a camera flash, the other one is not.

Using this equipment a dataset of videos captured while driving at night in a poorly lit environment has been collected. The algorithm presented in this chapter has been developed with this dataset in mind, something that will be reflected in the figures and results we will show from here on.

The videos has been captured while driving in a forested area and of great interest from a car safety perspective are the numerous animals present through out the video. Figure 3.1 shows a few unprocessed example frames from a typical video in the dataset.

The above gives us the full setting of the video data before enhancement. The camera produces at each time frame two parallel versions of a scene, a flashed and non-flashed one, contained in two frames (from here on called I_t^{f} and I_t^{n} for a given time frame t) of size 360×640. In the following section we will look at our approach for enhancing and fusing the frames received by the camera.

Figure 3.1: *Example frames. Each column shows the non-flash and flash frame of the same scene.*

3.2 Algorithm

We consider two video signals I^{n} and I^{f} which represent a non-flash and flash video respectively, captured in the manner described above. Both I^{n} and I^{f} are degraded by noise and non-uniformly illuminated. To gain a video of higher visual quality we suggest a procedure that processes both of these videos individually and finally merges them to produce a video of increased visual quality. It should be assumed that all image data is in a $[0, 1]$ double range.

The general idea is to, for each time step t, enhance both I_t^{n} and I_t^{f} as much as possible and thereafter fuse them in such a way that we gain both of their respective best features and lose their worst.

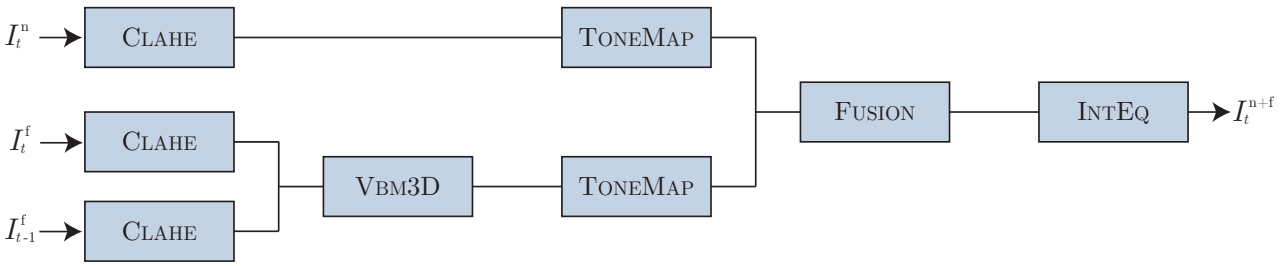


Figure 3.2: *The five step approach to video processing proposed here.*

The overall outline of this procedure can be seen in Figure 3.2 and is summarised briefly in pseudocode in Algorithm 3.1.

Algorithm 3.1: VIDEOSTREAMENHANCEMENT($I^{\mathrm{n}}, I^{\mathrm{f}}$)

$I_0^{\mathrm{f}} \leftarrow \textsc{Clahe}(I_0^{\mathrm{f}}, \text{params})$
initialise I^{avg} and I^{fm} (equation (3.1))

for all $t \geq 1$

do

Step 1:
$I_t^{\mathrm{n}} \leftarrow 0.5 \cdot \textsc{Clahe}(I_t^{\mathrm{n}}, \text{params}) + 0.5 \cdot I_t^{\mathrm{n}}$
$I_t^{\mathrm{f}} \leftarrow \textsc{Clahe}(I_t^{\mathrm{f}}, \text{params})$

Step 2:
$J_t^{\mathrm{f}} \leftarrow \textsc{Vbm3D}(I_{t-1}^{\mathrm{f}}, I_t^{\mathrm{f}})$

Step 3:
$I_t^{\mathrm{n}} \leftarrow \textsc{ToneMap}(I_t^{\mathrm{n}}, \text{params})$
$J_t^{\mathrm{f}} \leftarrow \textsc{ToneMap}(J_t^{\mathrm{f}}, \text{params})$

Step 4:
$I_t^{\mathrm{n+f}} \leftarrow \textsc{Fusion}(I_t^{\mathrm{n}}, J_t^{\mathrm{f}}, \text{params})$
update I^{avg} and I^{fm} (equations (3.2) and (3.3))

Step 5:
$I_t^{\mathrm{n+f}} \leftarrow 0.8 \cdot \textsc{IntEq}(I_t^{\mathrm{n+f}}, \text{params}) + 0.2 \cdot I_t^{\mathrm{n+f}}$

display $I_t^{\mathrm{n+f}}$

The algorithm consists of an initialisation and a for-loop with 5 different steps.

In the initialisation We start by performing CLAHE on the first flash frame and initialise two matrices, I^{avg} and I^{fm} that will later play a part in in the image fusion (FUSION) and intensity equalisation (INTEQ) steps. After that we run a series of five steps for all time steps $t \geq 1$:

Step 1: CLAHE is performed on both I_t^{n} and I_t^{f} to increase their local contrast.

Step 2: I_t^{f} is denoised using VBM3D with temporal window of size two (taking into account the previous frame I_{t-1}^{f})

Step 3: Logarithmic tone mapping is performed on I_t^{n} and I_t^{f}.

Step 4: The result of the enhancements of I_t^{n} and I_t^{f} frames are fused together.

Step 5: An intensity equalisation is performed that makes the tonal distribution of the frame more even.

3.2.1 Initalisation

We start of by initialising two images I^{avg} and I^{fm} which we name the *luminance matrix* and the *fusion matrix*.

$$I^{\text{avg}}(i,j) = I^{\text{fm}}(i,j) = 0.5, \quad \forall\,(i,j) \in [1,m] \times [1,n]. \tag{3.1}$$

These will later play a crucial part in the image fusion (FUSION) and intensity equalisation (INTEQ) steps.

3.2.2 Step 1: Contrast limited adaptive histogram equalisation

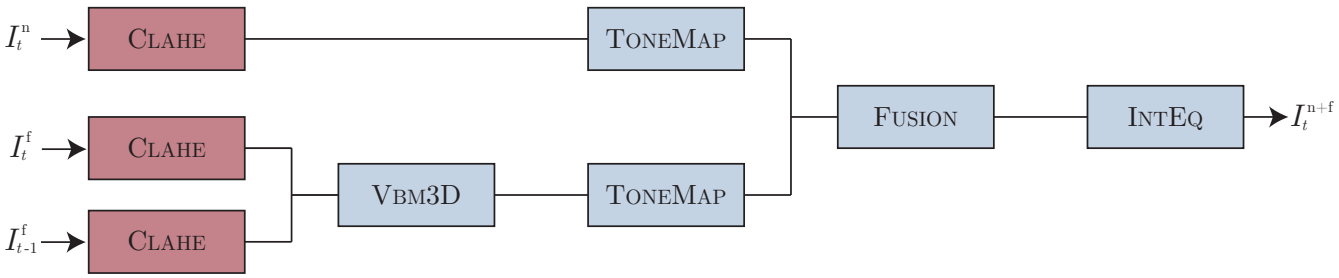


Figure 3.3: *Step 1, Contrast limited adaptive histogram equalisation (CLAHE).*

The purpose of the first step in our algorithm is to modify the contrast and the dynamic range of the frames I_t^{f} and I_t^{n}. For this we use the standard approach as proposed by [Zui94] and employed by the function ADAPTHISTEQ in Matlab's Image Processing Toolbox.

This is in accordance with what is described in Section 2.2.3, and the procedure can be further studied with the help of Algorithm 3.2.

Looking at the flowcharts or at Algorithm 3.1 we see that I_{t-1}^{f}, i.e. the flash frame from a previous time step, should be processed using this procedure as well. This is because we will be needing this frame in the upcoming denoising step.

Algorithm 3.2: CLAHE$(I, c_{\text{lim}}, [t_1, t_2])$

---

split image into $t_1 \times t_2$ tiles (see Figure 3.4)
for every tile $q \in \{q_1, q_2, ..., q_{t_1 \times t_2}\}$
do
- construct a histogram of 256 bins
- clip every bin that exceeds c_{lim} and redistribute uniformly over all other bins
- use the histogram to estimate a CDF
- employ this CDF as transformation function $f_q(x)$ for current tile q

for all $(i,j) \in [1,m] \times [1,n]$
do
- determine tiles q_k that are close to (i,j)
- determine weights c_{q_k} such that (i) is a linear interpolation
- $J(i,j) = \sum c_{q_k} \cdot f_{q_k}(I(i,j))$ (i)

return (J)

The CLAHE procedure is shown in Algorithm 3.2. The image is first split into tiles (see an example of this in Figure 3.4) and for each tile a CDF is computed. Each pixel transformation is after that determined by interpolation of CDF:s.

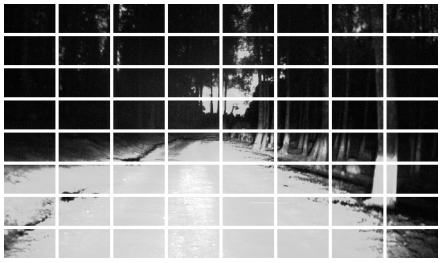


Figure 3.4: *Frame split into* 8×8 *tiles.*

3.2.3 Step 2: Video denoising using VBM3D

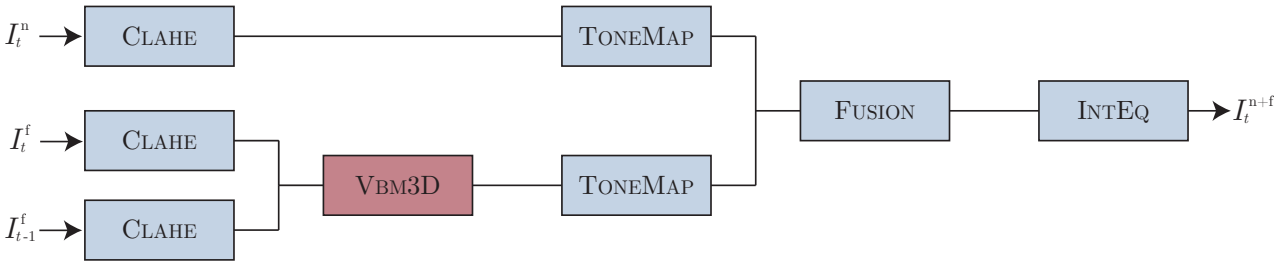


Figure 3.5: *Step 2, Video denoising (VBM3D).*

In the second step it it is time to denoise the frame I_t^{f}. We are employing video denoising using a temporal window of size 2, i.e. another frame I_{t-1}^{f} is taken into account in the way explained in Section 2.3.3.

This step only involves filtering the flash frame and not the non-flash frame. The reason for this is that high quality features of the non-flash frame are the ones illuminated by the headlights and they therefore contain relatively little noise. In the next section, covering the fusion step, we will cover how these features are used which should hopefully help explain this particular design choice.

Allthough there are several different video filtering algorithms one could imagine being used at this step, we use VBM3D because of its performance compared to other methods. The reader should be aware that all of the denoised results shown from here on in this report has been achieved with the help of VBM3D. This, however, does not imply that this algorithm can't work with any other method, and we don't rule out that there are cases in which other methods perform better. Nor do we rule out the possibility that other methods that outperforms VBM3D will be devised in the future.

3.2.4 Step 3: Logarithmic tone mapping

The third steps performs a logarithmic tone mapping on I_t^{n} and I_t^{f}. Recall from Section 2.2.2 that logarithmic tone mapping enhances contrast in dark regions of an image by transforming the pixel intensities using a logarithmic function. For an input image I (i.e. I_t^{n} or I_t^{f}) and

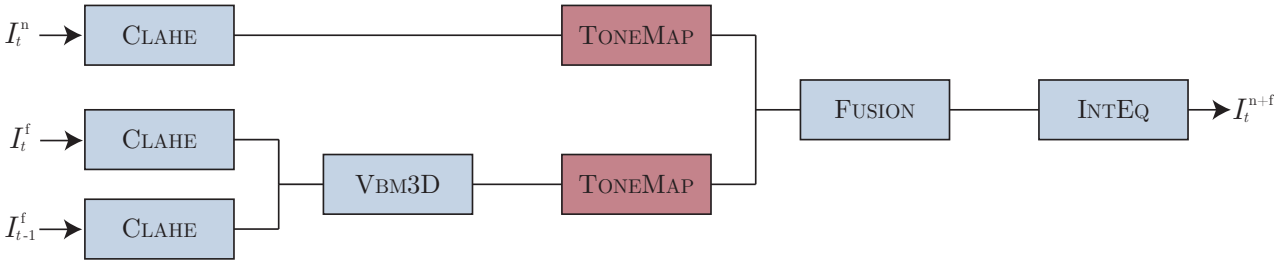


Figure 3.6: *Step 3, Logarithmic tone mapping.*

some parameter b that defines the steepness of the slope in the transformation function, the algorithm is:

Algorithm 3.3: TONEMAP(I, b)

---

for all $(i, j) \in [1, m] \times [1, n]$

do $\begin{cases} J(i,j) = \dfrac{\ln(10)}{\ln(256)} \cdot \dfrac{\ln(255 \cdot I(i,j) + 1)}{\ln\left(5 \cdot I(i,j)^{\frac{\ln(b)}{\ln(0.5)}} + 5\right)} \end{cases}$

return (J)

3.2.5 Step 4: Fusion

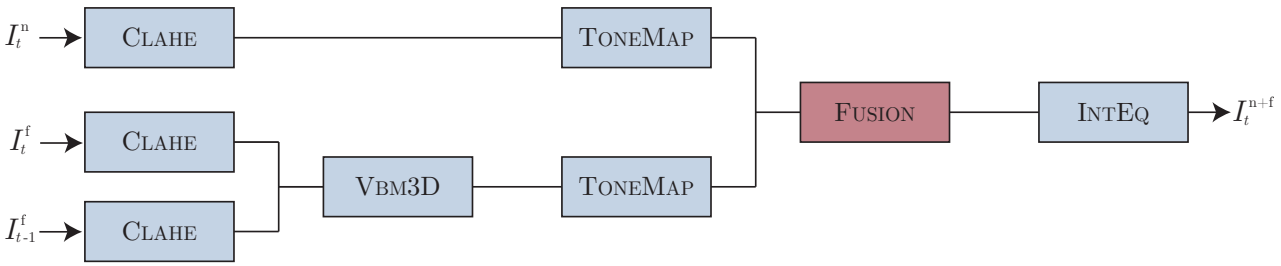


Figure 3.7: *Step 4, Fusion.*

The fusion step in our algorithm aims to combine I^{n} and I^{f} to bring out the best of both. I^{n} typically contains great detail in irradiated regions of an image, while I^{f} has better information in dark areas. The algorithm looks as follows:

Algorithm 3.4: FUSION$(I^{\mathrm{n}}, I^{\mathrm{f}}, I^{\mathrm{avg}}, I^{\mathrm{fm}})$

---

for all $(i, j) \in [1, m] \times [1, n]$

do $\begin{cases} w = 0.8 \cdot I^{\mathrm{fm}}(i,j) + 0.2 \cdot I^{\mathrm{avg}}(i,j) \\ I^{\mathrm{n+f}}(i,j) = w \cdot I^{\mathrm{n}}(i,j) + (1-w) \cdot I^{\mathrm{f}}(i,j) \end{cases}$

return $(I^{\mathrm{n+f}})$

As can been seen it employs the help of the fusion matrix I^{fm} and the luminance matrix I^{avg} to create a matrix of weights w which is used to create a combined version of I_t^{f} and I_t^{n}. Why we use a linear combination of both I^{fm} and I^{avg} will soon be explained.

For $t = 0$, I^{fm} and I^{avg} will simply be as they where initialised using equation (3.1). This means that every pixel in the fused image J will be the average of the corresponding pixels in I_t^{f} and I_t^{n}. Perhaps not ideal, but this will soon change as t increases and the movie feed progresses.

Updating I^{fm} and I^{avg}

To update I^{fm} we first update I^{avg}.

$$I^{\text{avg}} \leftarrow 0.9 \cdot I^{\text{avg}} + 0.1 \cdot \textsc{Smooth}(I_t^{\text{n+f}}, \sigma_{\text{avg}}) \tag{3.2}$$

where Smooth is a function that blurs the image using a Gaussian filter with standard deviation $\sigma_{\text{avg}} = 20$ in windows of 9×201 pixels. The width of this window is set to smooth well horizontally, i.e. keep horizontal edges and blur out vertical ones. The reason for this is the fact that we expect a low beam illumination, and aim to create a distinct edge between what is under the head lamps and what is not.

Using the above equation at every time step t, I^{avg} will in effect contain information of the current and historical tonal distribution of the video, given by the fact that take 9 parts of the old tonal distribution and 1 part of the new. This slow update of the tonal distribution ensures that I^{avg} can't change rapidly between two frames, thus preventing frame-to-frame flickering.

I^{avg} will by itself be used in the IntEq step, but right now we can use it to update I^{fm}.

At every time step t we construct a logical matrix I^{bw} with the help of I^{avg} . We remove small objects in it to get two coherent regions, one black (i.e. consisting of zeros only) and one white (i.e. consisting of ones only). Finally we add blurring for a smooth transition and update our fusion matrix I^{fm}. The procedure is:

$$I^{\text{bw}} = \begin{cases} 1 & \text{where } I^{\text{avg}} > c, \\ 0 & \text{where } I^{\text{avg}} \leq c, \end{cases} \tag{3.3a}$$

$$I^{\text{bw}} \leftarrow \textsc{FillHoles}(I^{\text{bw}}, N_{\text{nbh}}), \tag{3.3b}$$

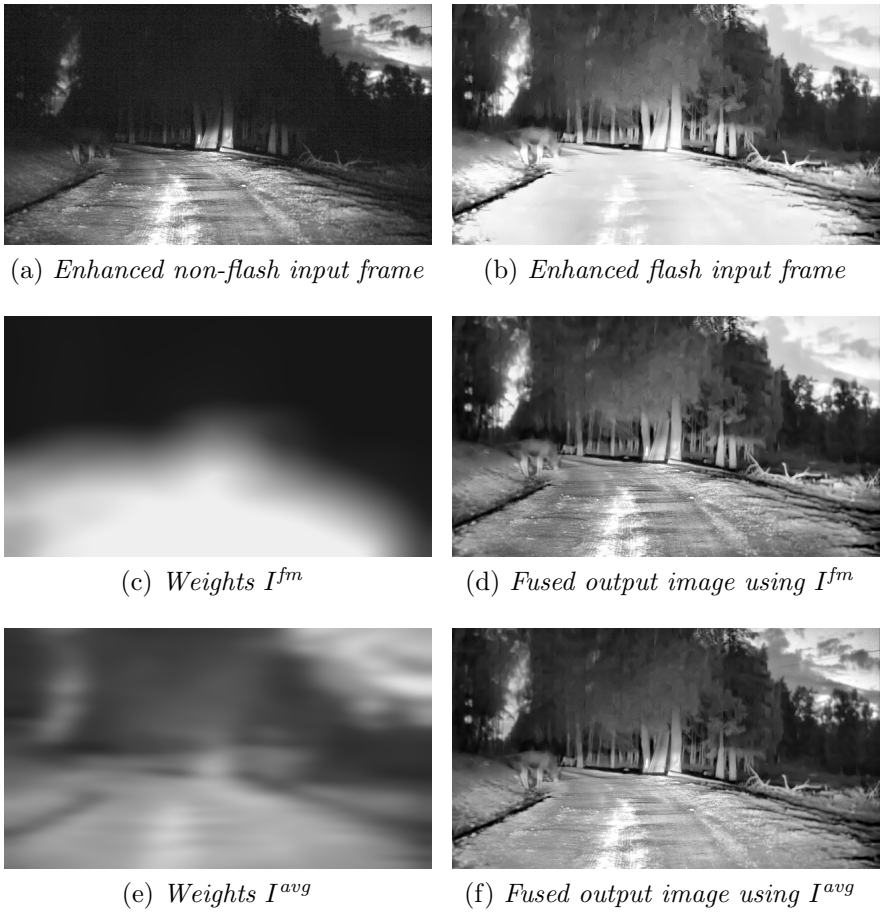
$$I^{\text{fm}} \leftarrow 0.9 \cdot I^{\text{fm}} + 0.1 \cdot \textsc{Smooth}(I^{\text{bw}}), \tag{3.3c}$$

for $c \in [0, 1]$. The parameter c can be chosen using for example the Matlab function graythresh which realises Otsu's Method [Ots75]. FillHoles removes smaller neighbourhoods of either zeros or ones. The parameter N_{nbh} specifies that all connected neighbourhoods with fewer than N_{nbh} pixels should be removed. We use $N_{\text{nbh}} = 30000$ which is roughly $1/7$ of the total number of pixels in a frame and using it we can be confident that we will be left with at least two big and plain neighbourhoods, one for the upper darker part and one for the lower.

After that, Smooth performs smoothing using a circular windowed averaging filter. Lastly, we update I^{fm} using 9 parts of its historical form and 1 part of the new calculations. Once again, the slow updating prevents frame-to-frame flickering.

It should be noted that one could completely use either I^{bw} or I^{avg} as weights in Algorithm 3.4, instead of a linear combination. An example of this is shown in Figure 3.8. Both methods has their merits as can be seen in Figure 3.8d and 3.8f. Using weights as in Figure 3.8c

produces a better separation between dark and light areas giving 3.8d a much nicer exposure of the road with better contrast and detail than in 3.8f. On the other hand Figure 3.8c misses that there are smaller light areas in the top portion of the scene and therefore makes the sky have less contrast in 3.8d. This is could in part be avoided by decreasing the parameter N_{nbh}, but doing so will also introduce flickering. This is why we opted for the midway approach with the linear combination in Algorithm3.4.



(a) *Enhanced non-flash input frame* (b) *Enhanced flash input frame*

(c) *Weights* I^{fm} (d) *Fused output image using* I^{fm}

(e) *Weights* I^{avg} (f) *Fused output image using* I^{avg}

Figure 3.8: *Fusion weights comparison.*

3.2.6 Step 5: Intensity equalisation

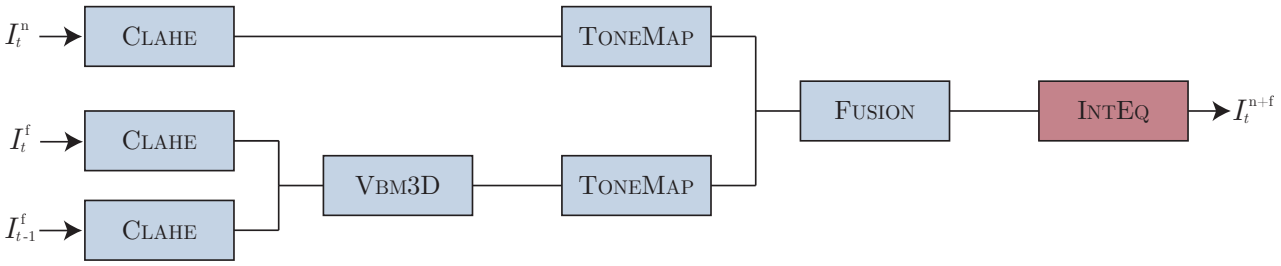


Figure 3.9: *Step 5, Intensity equalisation.*

After having produced a fused frame we arrive at the final step where it is time to deal with issues of non-uniform illumination. In the current application the observed scene is captured at night from a moving vehicle with its headlights, normally, turned on. This gives us the effect that the area where the headlights strike have higher intensity than areas not reached by the light. This in turn gives this the unwanted effect that the observed scene appears to be divided in two parts, one dark and one light. See Figure 3.10a for an example of this. What we want to do is to normalise the intensities of the frames in such a way that the average intensities in different areas of the frame are roughly the same. This could to some extent be accomplished by CLAHE using a higher clip limit. However, we have seen that a higher clip limit magnifies noise too much.

(a) *Before intensity equalisation* (b) *After intensity equalisation*

Figure 3.10

In pseudocode, the intensity equalisation goes as follows:

Algorithm 3.5: $\textsc{IntEq}(I, I^{\text{avg}})$

for all $(i,j) \in [1,m] \times [1,n]$
do $\left\{ J(i,j) = \dfrac{I(i,j)}{I^{\text{avg}}(i,j)} \right.$

$J_{\min} = \textsc{Min}(J)$
$J_{\max} = \textsc{Max}(J)$
if $J_{\min} < J_{\max}$
do $\left\{ \begin{array}{l} \textbf{for all } (i,j) \in [1,m] \times [1,n] \\ \quad \textbf{do } \left\{ J(i,j) \leftarrow \dfrac{J(i,j) - J_{\min}}{J_{\max} - J_{\min}} \right. \end{array} \right.$
return (J)

MIN and MAX output the minimum and maximum value of a matrix respectively, and are used to adjust the elements in J so they are in $[0, 1]$.

3.3 Implementation

Throughout this thesis work, all implementations have been carried out in Matlab. Matlab is a commonly used tool within the digital processing and analysis field and many researchers do in fact share Matlab software packages that accompanies their published research.

One of the shortcomings using Matlab can sometimes be the lack of speed. However, our aim has solely been to create the algorithmic framework, in real-world applications the processing should be carried out by custom design hardware.

3.3.1 Image processing toolbox

Although basic Matlab has many basic built-in functions for handling image data we have found it convenient to also the add-on package called the *Image Processing Toolbox*. The toolbox comes with an easier to use set of visualisation functions and with a better support for different image file formats. Further, it also has a wide set of reference-standard algorithms, i.e. the ADAPTHISTEQ function that realises the CLAHE method previously discussed in this report.

3.3.2 Useful software packages

For the filtering step we have used a software package implementing the BM3D and VBM3D algorithms available at [Dan+12]. VBM3D is the most complex part the algorithm we propose, it is a Matlab function that utilises the MEX-functionallity. This means that bottleneck computations are executed in C/C++ which greatly increases the efficiency.

3.3.3 Alternative frameworks

There are of course alternatives to using Matlab for the kind of work that we have been doing. For example, C/C++ implementations of published algorithms are very commonly seen. In a project where time consumption is an issue right from the very start, working in C/C++ might in fact be the wisest choice.

Worth mentioning is also the OpenCV (Open source Computer Vision) framework, which is released under the BSD license and free for use for both academic and commercial purposes.

3.3.4 Time consumption and future implementations

We mentioned at the very start of this report that realtime processing speeds are not something that we have been aiming at during our work. Nevertheless, timing issues are a factor, mainly for two reasons.

Work like this involves a lot of testing and parameter tuning and that means that the speed of the system effects how exhaustively one is able to test different set ups. That has been one only slight drawback we have experienced when working with Matlab, and from that point of view another setting might have been slightly better.

For future PC-based testing of our work it could potentially be wise to consider either working in a different programming language, look at the possibility to run certain parts of the code using the GPU, or to simply use a more powerful computer.

The second reason why timing is a factor is due to the fact that the general aim is that our work should evolve to something that can achieve realtime speed when implemented on specialised hardware. If what we presented here had no chance of achieve such speeds it would undoubtedly be less interesting. Luckily, we don't believe that that is the case.

In Table 3.1 we see a layout of the speed of our current implementation of the full algorithm presented in the previous chapter.

Table 3.1: Time consumption for the different steps.

Step	*sec/frame*
TONEMAP + CLAHE	0.3
VBM3D time	0.5
FUSION time	0.8
INT-EQ time	0.05
Overhead time	0.6
Total time	2.2

For the time consumption to be measured we have used MATLAB 2011b on a Windows computer with an Intel Core 2Duo 3.00 GHz processor and 3 GB of RAM. A total of roughly 2 seconds processing time per produced frame might be a slight hassle when running the algorithm on long video sequences but with regards to achieving realtime speeds on specialised hardware we feel confident that it would be possible.

3.4 Related and alternative approaches

Our method has partly be inspired by [Xu+10], in which the authors present a method for enhancing very dark videos. Their approach is divided into three step. They first perform a pre-denoising of the video, then a logarithmic tone mapping, as found in [Dra+03], and finally a post-denoising. They use 3D NLM (3D Non-Local Means) as their denosing method, with an alteration of it in the pre-denoising step. The alteration they claim prevents them from gaining ghosting artifacts, something that the original 3D NLM is prone to give.

It is certainly possible to adapt our approach to such a two-step filtering scheme and it might in some settings be useful. Exchanging one denoising step, with potentially aggressive denoising, for two, lighter denoising steps where the first one is divided into spatial and temporal denoising (the results are combined by weights depending on the motion content) could prevent artifacts such as ghosting. When using VBM3D we have, however, not experienced that there is any risk of ghosting (with our current data), and theoretically denoising can be done both before and after tone mapping as long as the denoising strength is set accordingly.

In [Mal+07], another paper with work in the same area as ours, the authors present a general method for enhancing noisy and low contrast videos that is inspired by the research of vision in nocturnal animals. As their denoising approach they have opted to use is the structure-adaptive anisotropic filter presented in [Yan+96] and for tone mapping they use a modified version of CLAHE. They do not however work with parallel video feeds the way we do.

Work with parallel video feeds can be found in [BMM07], where a method to enhance

underexposed video data using fusion between simultaneously captured visible and nonvisible spectrum data is presented. Their approach is related to ours in the sense that we also fuse to data captured in different ways, although our data is not *multispectral* but rather *multiexposed*. The approach in [BMM07] is also more involved and relies on that the nonvisible video data is more or less free of noise. Further their approach requires ego-motion estimation to be able to deal with a moving camera sequence.

4 Results

To objectively judge the quality of a video acquired from some video processing system is not an easy task. The most traditional way of doing this in the literature is by calculating the so called *signal-to-noise ratio* (SNR) or *peak signal-to-noise ratio* (PSNR). The SNR and PSNR are calculated for a processed video by looking at each processed frame and comparing it to a corresponding *true* frame in a *true* video, i.e one not degraded by noise and other irregularities. This of course can only be done in a setting where one has a true and undistorted video to begin with, as is the case for example in a video compression system. In our case however, we only have distorted data due to the very nature of the problem we are working with. Further, noise is only half the issue at hand here, and we are as much interested in improvement of illumination as we are in noise reduction. The bottom line is that although highly desirable, an objective metric for evaluating the visual quality of our processed videos seem to be out of the picture.

A second approach is to evaluate processed videos subjectively. One idea to do this is by a so called *mean opinion score* (MOS) that stems from the telephone network industry. In this a test group of listeners (in our case they would be viewers) individually score the quality of a call (or video) using a scale of 1 to 5 which is then averaged. However, even though subjective testing like this would be interesting it is unfortunately beyond the scope of our work.

In this chapter we will instead evaluate the algorithm found in Chapter 3 by presenting a series of test cases that aim to show the advantages and disadvantages of certain design choices. By leaving out certain steps in our algorithm in the test cases we hope that the reader should gain a better understanding of what these steps do and why we chose to include them in the first place.

4.1 Test cases

We will start by looking at the results we gain if we run the algorithm exactly as presented in the previous chapter.

The video we will be looking at contains a scene of a graveled road at night with animals occasionally standing at the road side. The effects we are looking for after processing is primarily noise levels, if important objects like animals can be seen and how even the illumination level is spatially across a scene.

For noise removal, where we are temporally looking at two frames, the current and the previous. A full presentation of all the design parameters can be found in Appendix A.

The result can be seen in Figure 4.1. The figure is ordered column-wise with one scene in each column and in each row the non-flash, flash and output frame of each scene.

First of all, the resulting frames in the bottom row have a strikingly more even illumination than the unprocessed input frames which is just what we wanted. No part of the output frames are heavily under- or overexposed.

As far as visibility goes from a car safety perspective, the most important objects are the animals present in all three scenes. In the non-flash frames these are more or less invisible in all three cases, and in the flash frames only slightly more visible. The output frames in the bottom row shows a drastic improvement compared to both, notice in particular the animal on the right side of the road in the rightmost scene that is quite clearly visible in the output

Figure 4.1: *Full approach. The columns contains two different scenes and the rows the non-flash, flash and output frame of each scene respectively.*

frame.

Figure 4.4 shows a zoom-in that compares at a more detailed level the improvement achieved in the output frame compared to the input data.

There is still some noise present in the output frames especially in the areas of the trees in the scenes. Looking at an individual output frame is however one thing since the presence of noise between frames might be a more important matter. Unfortunately it is not possible to show the visual improvement of the in-between frame noise in a report like this but we can use a measurement to give the reader an indication that this is true. If one picks a homogeneous spatial region without any obvious objects in it (like a region in the trees in the left most scene in Figure 4.1) and compare the change of that region between two frames one can get an indication of the temporal noise. Since the region is homogenous and has no objects in it (in either frame) there should be little change if there is little temporal noise. To measure the change we can use a correlation coefficient given by

$$c_l = \frac{\sum_m \sum_n (I_t^r(m,n) - \overline{I_t^r}) \cdot (I_{t+l}^r(m,n) - \overline{I_{t+l}^r}))}{\sqrt{\sum_m \sum_n (I_t^r(m,n) - \overline{I_t^r})^2 \cdot \sum_m \sum_n (I_{t+l}^r(m,n) - \overline{I_{t+l}^r}))^2}} \tag{4.1}$$

where I_t^r notates a region r in a frame I_t and $\overline{I_t^r}$ its mean.

Figure 4.2 shows two frames from a video, 10 temporal steps apart from each other. There are also four regions shown in the figure and as can be seen these don't change very much in information content in the course of a ten frame progression, besides being slightly brighter in the last frame.

Using equation (4.1) we can calculate ten correlation estimates. One between frame 1 (shown in Figure 4.2) and frame 2, one between frame 2 and frame 3 etc. up to frame 11. The result of these calculations are shown in Figure 4.3 together with corresponding calculations for the same regions in the flash input frames. The circle-marked line shows the correlation of the processed frames and the square-marked line the correlation of the flash input frames, and as can be seen the general picture is that the correlation roughly doubles after processing.

(a) *Frame 1* (b) *Frame 10*

Figure 4.2: *Regions selected for temporal correlation.*

We once again remind the reader that the correlation calculation is a very simplistic estimate for temporal noise but nevertheless, the results shown are as one would expect. The sceptic reader should instead take a look at the videos **flashvideo.avi** and **enhancedvideo.avi** where the reduction of noise is very visually apparent (see Appendix A for information on how to acquire the videos).

With these results in mind we will go on to look at what steps in the algorithm produce what results.

4.1.1 Test case 1: An approach without CLAHE

The first thing we will look at is what happens if we remove the first step, namely CLAHE. This means that for changes in greyscale level distribution, i.e illumination levels, only logarithmic tone mapping and intensity adjustment (i.e step 3 and 5 in the algorithm) will be used.

In Figure 4.5 we see three different result frames, (from top to bottom) one acquired without contrast enhancement, one with a simple global histogram equalisation (HE) and one with CLAHE. First of all the results from using HE is obviously not good and we can see that the noise in the data has been seriously over-amplified. The result from not using any histogram equalisation do at first glance look fairly comparable to the original result with CLAHE. The road can be seen to have better structure using CLAHE but the result without it does in fact even look slightly smoother. But if we take a look at some zoomed-in portions of the resulting frames shown in Figure 4.6 we do see a difference where the advantage of CLAHE can be seen

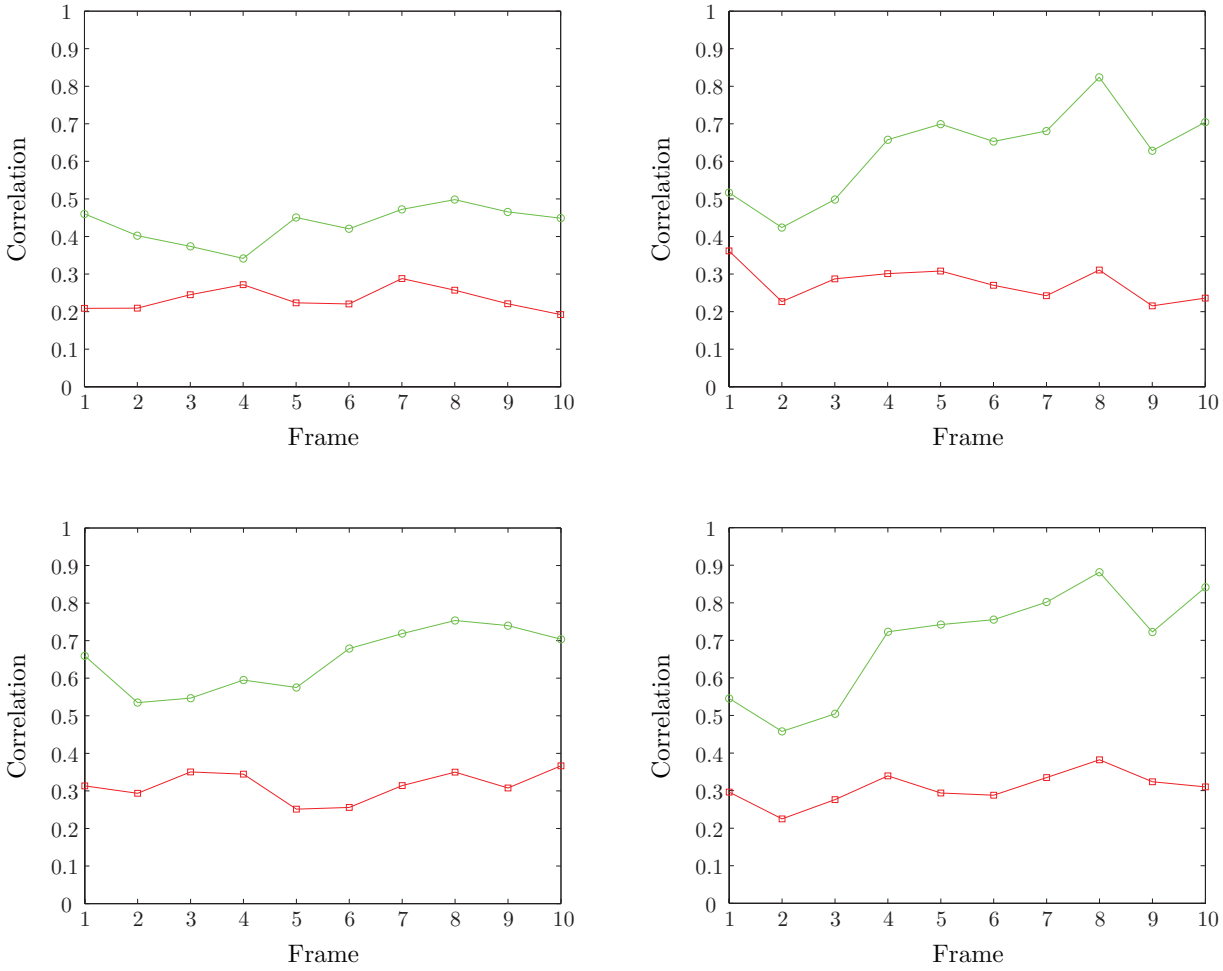


Figure 4.3: *Temporal correlation in four different regions. Lines marked with* □ *correspond to flash inputs, while lines marked with* ○ *correspond to processed results.*

as it gives better contrast and details and a greater sense of depth. This effect becomes even more apparent in a moving video.

4.1.2 Test case 2: An approach without any denoising

One of the more computationally heavy parts of our approach is the video denoising step and it is therefore important to explore if filtering is justified in our particular case. Indeed, it would be very beneficial if this step could be left out all together since that would greatly lower the computational complexity of the algorithm.

Unfortuantely, we see in Figure 4.7 that by leaving out the filtering we get a result with a considerably higher level of visible noise. The zoom-ins in Figure 4.8 confirms this further. The conclusion to be drawn is that if one wants to do the kind of light adjustments that we are doing, some sort of denoising step is indeed needed.

Figure 4.4: *Zoom-ins. Top: non-flash, flash. Bottom: output.*

(a) *No histogram equalisation*

(b) *Global histogram equalisation*

(c) *CLAHE*

Figure 4.5: *Histogram equalisation comparison.*

4.1.3 Test case 3: The effect of logarithmic tonemapping

The simplest approach to enhance the visual quality of the video is arguably to just transform it using a logarithmic tone mapping function. As stated in Section 2.2.2 this has the effect of widening the dynamic range of the darker grey-scale levels of the image. But since we are also

Figure 4.6: *Zoom-in comparison. To the left: a zoom-in of a result without histogram equalisation. To the right: result with CLAHE.*

Figure 4.7: *Comparison between a denoised and non-denoised output.*

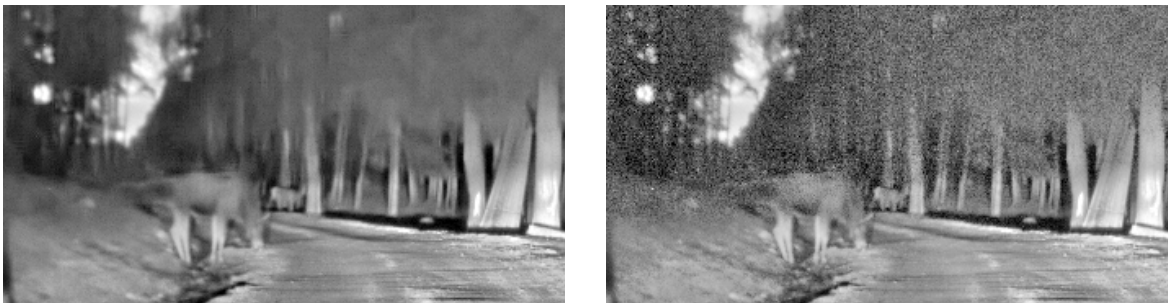


Figure 4.8: *Zoom-in comparison between a denoised and non-denoised output.*

using CLAHE which is also in effect a tonemapping procedure one could argue that adding logarithmic tone mapping is unnecessary.

In Figure 4.9 we see a comparison between using and not using logarithmic tone mapping in our algorithm. We can see that CLAHE and intensity equalisation do a pretty good job at uniforming the illumination levels across the scene. However, once again looking at still frames do not convey the full story.

The interested reader can have a look at the video **enhancedvideo-notonemap.avi**, where it is more apparent that when leaving out logarithmic tone mapping we are left with a video where it is more obvious that upper regions of the video are darker than the lower, i.e with less uniform illumination. An effect which is undesirable.

(a) *Without logarithmic tonemapping* (b) *With logarithmic tonemapping*

Figure 4.9: *The effect of logarithmic tone mapping*

4.1.4 Test case 4: An approach without intensity equalisation

The final step in our proposed algorithm is an intensity equalisation step with the aim of disguising the fact that in the original data only the lower half of the scene in a given frame is illuminated by the headlights of the car.

In the bottom row of Figure 4.10 we see what happens if we turn of this feature. Remember that the intensity equalisation method we use has a temporal component, that is why we see a very little change in the leftmost scene, which is at the very start of the video, compared to the middle and rightmost scene. This means that in practice, the system needs a few seconds to adjust itself after start. In the latter scene the disadvantaged of not having an intensity equalisation step is the most striking, since the eye is drawn to light it might be that at a quick glance one will pay attention to the illuminated road and not notice that there is an animal standing right at the end of the headlights reach.

Figure 4.10: *The effect of intensity equalisation. One scene in each column, intensity equalisation only performed for the top row.*

But what happens if we increase the effect of the CLAHE step? It might seem that doing so should also equalise the illumination levels across the scene and therefore render this step

unnecessary. To some extent that is also true but there are two drawbacks with that approach.

First of all, the more the clip limit of CLAHE is increased, the more the noise present will increase. This can to some extent be taken care of by changing the parameters of the chosen denoising algorithm to increase the level of denoising. Depending on the denoising algorithm this might work if it is the case that the algorithm can perform heavier smoothing without also blurring the edges.

Secondly, and more importantly, it is harder to get rid of the *edge* that exists between where the more illuminated area meets the darker area. To CLAHE that edge is no less an edge than any real physical edge present in the scene and therefore the contrast stretching might have the opposite effect and make it more visible.

4.1.5 Test case 5: An approach without fusion

Before arriving at the fusion step in the proposed algorithm, both the non-flash and flash frame undergo individual enhancement. It could be the case that one or the other of these frames are of sufficient quality, thus rendering the fusion step unnecessary.

A very simple enhancement method that can be done on an individual frame is logarithmic tonemapping.

Figure 4.11: *Effect of only applying logarithmic tone mapping. Using* $b = 3.75$ *for non-flash,* $b = 2.5$ *for flash*

This is shown in Figure 4.11 and and even though the stretching of the grey-scale distribution helps reveal information, both frames end up being much to noisy and large areas are under or overexposed. The parameter b could be increased (giving a less step transformation function, i.e less stretch on the darker greyscale level) but that would decrease the the contrast in the darker areas, thus defeating the purpose.

Moving on, we can instead try to apply all steps but the fusion of our algorithm and see what happens to each feed individually. In the full approach with fusion we have seen that it is unnecessary to denoise the non-flash frames, since in that case the information we take from it is mainly from its lower parts, but now when we plan to use the whole of each non-flash frame this is something we have to do.

Figures 4.12 and 4.13 shows what happens when we perform CLAHE, video-denoising, logarithmic tone mapping and intensity equalisation to each of the two feeds. Its easy to distinguish that the non-flash feed has better qualities in the lower parts of the scene compared to the flash feed, and vice versa. We can confidently draw the conclusion that by employing fusion we can gain the best of both video feeds.

(a) *Non-flash video individually enhanced without fusion*

(b) *Flash video individually enhanced without fusion*

(c) *Standard approach with fusion*

Figure 4.12: *Individual enhancement of both video feeds without fusion.*

(a) *Zoom-in of 4.12a*

(b) *Zoom-in of 4.12b*

(c) *Zoom-in of 4.12c*

Figure 4.13: *Zoom-ins of Figure 4.12.*

4.2 Further enhancement

If we have a look at Figure 4.14 and 4.15 we see an example of a scene where an animal, i.e a potential danger, has been further enhanced compared to the output of our algorithm. In this case however this has been done manually by selecting a region for enhancement and adding an additional CLAHE and denoising step. This purely aims to show what could be done if our system was matched with a system for object detection. Using such a system, objects of potential danger can be highlighted, for example as seen in the figure, potentially also by pseudo colouring the object or while lowering the contrast of everything else.

Figure 4.14: *The rightmost image has had one of the animals in the scene selectively enhanced.*

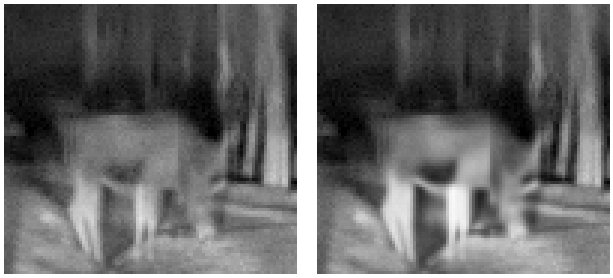


Figure 4.15: *Zoom-in of Figure 4.14.*

5 Discussion

In this report we have presented an automatic method for enhancing the visual quality of noise degraded videos with poor or non-uniformly illuminated scenes. The work presented has been presented in a proof-of-principle manner and we believe it gives a fair view of the result one can expect to achieve if this approach was to be implemented in a hardware-based realtime system.

We imagine that the method we have devised could be part of a bigger vehicle safety system, a system that could actually and intelligently detect dangers in front of the vehicle and properly alert the driver. We have discussed the possibility to incorporate our approach within object/danger detection setting to gain information that enables the system to know which regions of a scene to enhance the most. We do believe that something like this would be necessary for the system to be able to work well from a car safety perspective.

How to present the information from a car safety system to the driver in the best way is practically a field of science in itself. The dream of every interaction designer seems to be the idea to project this kind of information on to the actual windscreen of the car, possibly such that objects that are a potential danger can be lit up in such a way that it is lined up with the field of vision of the driver and the real world object that has been judged to be of danger.

As far as further work goes, our approach would benefit from further testing. We have discussed objective testing of visual enhancement testing with PSNR scores etc. One possibility to do this could be by obtaining data with high visual quality which can be artificially degraded in some (preferably realistic) way.

We have also mentioned subjective testing as a potential benefit for this kinds of work. How to do this usefully is a matter of discussion. We do believe that for this to be truly useful the testing should take place in a driving simulation environment, since it is more interesting to see how well a person preoccupied by driving is able to spot dangers with our system rather then a person that don't have to pay attention to anything else.

Finally, it would be beneficial to add more adaptiveness to our approach. To make it more robust it could for example be helpful if the system could be fed with external information such as the level of exposure, preferably at different points in the captured scene, and noise levels. This would make the system more tightly coupled with the camera equipment, something which in itself would be an interesting continuation of our work.

A Parameters

To produce the images found in Chapter 4 of this report the following parameters have been used.

- Clahe

  $c_{\text{lim}} = 0.01$ in Algorithm 3.2 for both I^{f} and I^{n}

- Vbm3D

  $\sigma = 20$

- ToneMap

  $b = 3.75$ for I^{n} and $b = 2.5$ for I^{f} in Algorithm 3.3. The exact setting of b isn't highly crucial but we have followed the advise from [Dra+03] and opted for a setting that gives rather low mapping functions for low intensities.

- Updating I^{fm} and I^{avg}

  $\sigma_{\text{avg}} = 20$
  $N_{\text{nbh}} = 30000$

To acquire the videos **flashvideo.avi**, **darkvideo.avi** and **enhancedvideo.avi** send an email to either *johan.karlsson@fcc.chalmers.se* or *mats.kvarnstrom@fcc.chalmers.se*.

References

[BCM+05] A. Buades, B. Coll, J. Morel, et al. "A review of image denoising algorithms, with a new one". In: *SIAM Journal on Multiscale Modeling and Simulation* 4.2 (2005), pp. 490–530.

[BM05] E. Bennett and L. McMillan. "Video enhancement using per-pixel virtual exposures". In: *ACM Transactions on Graphics (TOG)*. Vol. 24. 3. ACM. 2005, pp. 845–852.

[BMM07] E. Bennett, J. Mason, and L. McMillan. "Multispectral bilateral video fusion". In: *Image Processing, IEEE Transactions on* 16.5 (2007), pp. 1185–1194.

[Bov05] A. Bovik. *Handbook of image and video processing*. Academic Press, 2005.

[Dab+07] K. Dabov et al. "Image denoising by sparse 3-D transform-domain collaborative filtering". In: *Image Processing, IEEE Transactions on* 16.8 (2007), pp. 2080–2095.

[Dan+12] A. Danielyan et al. *Image and video denoising by sparse 3D transform-domain collaborative filtering*. June 2012. URL: `http://www.cs.tut.fi/~foi/GCF-BM3D/`.

[DFE07] K. Dabov, A. Foi, and K. Egiazarian. "Video denoising by sparse 3D transform-domain collaborative filtering". In: *Proc. 15th European Signal Processing Conference*. 2007.

[Dra+03] F. Drago et al. "Adaptive logarithmic mapping for displaying high contrast scenes". In: *Computer Graphics Forum*. Vol. 22. 3. Wiley Online Library. 2003, pp. 419–426.

[GTC02] S. Gutta, M. Trajkovic, and A. Colmenarez. *Vehicular blind spot vision system*. US Patent 6,424,272. 2002.

[LDW09] S. Lansel, D. Donoho, and T. Weissman. "DenoiseLab: a standard test set and evaluation method to compare denoising algorithms". In: *Benchmarks and MATLAB software online at {http://www. stanford. edu/˜ slansel/DenoiseLab/}* (2009).

[Mal+07] H. Malm et al. "Adaptive enhancement and noise reduction in very low light-level video". In: *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*. IEEE. 2007, pp. 1–8.

[Mol+08] A. Molinero et al. *TRACE Project. Deliverable 1.3. Road users and accident causation. Part 3: Summary report*. Anglais. Rapport de recherche. 2008. URL: `http://hal.archives-ouvertes.fr/hal-00545396`.

[Ots75] N. Otsu. "A threshold selection method from gray-level histograms". In: *Automatica* 11 (1975), pp. 285–296.

[Por+03] J. Portilla et al. "Image denoising using scale mixtures of Gaussians in the wavelet domain". In: *Image Processing, IEEE Transactions on* 12.11 (2003), pp. 1338–1351.

[RC12] Y. Rao and L. Chen. "A survey of video enhancement techniques". In: *Journal of Information Hiding and Multimedia Signal Processing* 3.1 (2012), pp. 71–99.

[SA96] E. Simoncelli and E. Adelson. "Noise removal via Bayesian wavelet coring". In: *Image Processing, 1996. Proceedings., International Conference on*. Vol. 1. IEEE. 1996, pp. 379–382.

[TM98] C. Tomasi and R. Manduchi. "Bilateral filtering for gray and color images". In: *Computer Vision, 1998. Sixth International Conference on*. IEEE. 1998, pp. 839–846.

[Xu+10] Q. Xu et al. “A new approach for very dark video denoising and enhancement”. In: *Proc. 17th International Conference on Image Processing*. IEEE. 2010, pp. 1185–1188.

[Yan+96] G. Yang et al. “Structure adaptive anisotropic image filtering”. In: *Image and Vision Computing* 14.2 (1996), pp. 135–145.

[Zui94] K. Zuiderveld. “Contrast limited adaptive histogram equalization”. In: *Graphics Gems IV*. Academic Press Professional, Inc. 1994, pp. 474–485.