MULTI-SCALE VIDEO ENHANCEMENT
Artifact reduction and contrast boosting

M.L. Poelstra

Supervisor: prof.dr.ir. G. de Haan
Coach: dr.ir. H. van der Heijden (Philips Research Laboratory)
Date: November 2005

The Faculty of Electrical Engineering of the Eindhoven University of Technology does not accept any responsibility regarding the contents of Master’s Theses
Concerns: Internship Report
Period of Work: May-October 2005
Notebooks: None

Authors' address M.L. Poelstra m.l.poelstra@student.tue.nl
Abstract: Modern day television sets are increasingly using LCD or PDP technology. One of the concerns with these technologies lies in the more prominent visibility of artifacts, such as blocks from MPEG-encoding. As digital encoding and transmission is emerging, this requires proper action to be taken to improve the (perception of) video. Additionally, many people want the video to be as 'sharp' as possible, which is usually realised by post-processing of the data. Given the rise of the digital MPEG-chain, this would imply an even more prominent visibility of the (coding) artifacts; a second concern. This project uses tools from multi-scale image processing to create an integrated artifact reduction and contrast boosting algorithm. External metrics can be used to tune the filter settings.

Conclusions: The multi-scale approach has promising results: it is possible to have combined artifact reduction and contrast boosting in one algorithm, for both low and high quality video.

Key parameters for picture quality (to tune the amount of artifact reduction and/or contrast boosting) are determined, and a Graphical User Interface is provided to tune these settings while processing a 'live' video stream. Also, 'High Scale Motion Detection' was developed, assisting in the automatic tuning of filter settings. Other metrics (like a block grid detector) can be included as well, and are needed for automatic adjustment to different kinds of input content, e.g. high quality DVD and low quality DivX material.
## Contents

1 Introduction .......................................................... 1
   1.1 Structure of this report ...................................... 1
   1.2 Internship assignment ........................................ 2

2 Introduction to multi-scale image enhancement ................. 3
   2.1 Laplacian Pyramid ............................................ 4
   2.2 Contrast boosting ............................................ 6
      2.2.1 Definition and caveats ................................ 6
      2.2.2 Preventing noise amplification ....................... 7
      2.2.3 Preventing halos ........................................ 7
      2.2.4 Results ................................................ 9
   2.3 Noise reduction .............................................. 9
      2.3.1 Definition and caveats ................................ 9
      2.3.2 Implementation ......................................... 11
      2.3.3 Eliminating CATS ...................................... 11

3 Enhancements .................................................................. 15
   3.1 Graphical User Interface (GUI) ............................... 15
   3.2 Full-size approach ............................................ 17
   3.3 Combining boosting and reduction ........................... 18
      3.3.1 Noise reduction .......................................... 18
      3.3.2 CATS revisited ........................................... 19
      3.3.3 Integrated boosting and reduction .................... 21
   3.4 Content adaptive filtering .................................... 23
      3.4.1 High Scale Motion Detection (HSMD) .................. 24
      3.4.2 Other metrics ............................................ 26

4 Evaluation .................................................................. 29
   4.1 Full-size versus Sub-sampled ............................... 29
   4.2 Effects of parameter changes ................................. 30
      4.2.1 Sharpness .................................................. 30
      4.2.2 CircleRadius ............................................. 30
      4.2.3 CircleShape ............................................... 35
      4.2.4 MaxGain .................................................... 35
      4.2.5 MaxCATS .................................................. 35
      4.2.6 MotionSensitivity ....................................... 35

© Koninklijke Philips Electronics N.V. 2005
4.3 Typical examples ................................................. 35
4.4 Efficiency of integrated algorithm ............................ 40

5 Conclusions ......................................................... 41
5.1 Answers to the Research Questions ............................ 41
5.1.1 Can both noise reduction and contrast boosting be combined in one algorithm? ........................................... 41
5.1.2 Can external indicators help to improve the output? ........ 42
5.1.3 Is it possible to efficiently detect 'motion' at the highest scale and tune the filter with it? ...................................... 42
5.1.4 Does a full-size approach improve image quality? .......... 42

5.2 Discussion ......................................................... 42
5.3 Suggestions for future work ...................................... 43
5.3.1 Metrics ......................................................... 43
5.3.2 Colour ......................................................... 43
5.3.3 Performance ................................................... 43
5.3.4 Training ......................................................... 44
5.3.5 Subjective Assessment ........................................ 44

References .......................................................... 45
Section 1

Introduction

Modern day television sets are increasingly using LCD or PDP technology. One of the concerns with these technologies lies in the more prominent visibility of artifacts, such as blocks from MPEG-encoding. As digital encoding and transmission is emerging, this requires proper action to be taken to improve the (perception of) video. Additionally, many people want the video to be as 'sharp' as possible, which is usually realised by post-processing of the data. Given the rise of the digital MPEG-chain, this would imply an even more prominent visibility of the (coding) artifacts; a second concern.

This calls for an algorithm that handles both contrast/sharpness enhancement and artifact/noise reduction, although the two contradict each other when simply cascaded. Therefore, special measures must be taken to boost some parts of the video, while suppressing other parts.

The approach taken in this project uses tools from multi-scale image processing, which has some interesting properties over other methods. A great advantage is the possibility to tune frequency components in the image using information in the spatial domain. This property can also be used to apply stronger filtering on parts of the image, e.g. parts that changed much due to motion.

1.1 Structure of this report

In the preamble of this section, the project is put in a social context and a glimpse of the project-goal is stated. Next, the problem-statement of the project will be expressed as research questions.

Before those questions can be answered, a short introduction to the multi-scale approach will be given in section 2, including preliminary work.

The additions resulting from this internship are documented separately in section 3.
The improvements of section 3 are evaluated in section 4.

Finally, section 5 gives conclusions and provides thoughts on how the algorithm may further be improved.

1.2 Internship assignment

The research questions for this (three months) internship are expressed as:

1. *Can both noise reduction and contrast boosting be combined in a single algorithm, using the multi-scale approach?*
   Noise reduction can be characterised as the process of removing (mostly) high-frequency components from the image. Conversely, contrast boosting amplifies frequency components in the image. When simply combining these enhancement techniques, the effect of one filter will (at least partly) cancel the effect of the other. There might be ways to integrate both filters and adapt them to the local picture content such that noise is reduced, but *low contrast details* are boosted.

2. *Can external indicators (like block-grid detectors, noise detectors) help to improve the output?*
   For example, knowledge about the position and visibility of block-edges in the image might prevent the boosting of those edges and may even be used to suppress them.

3. *Is it possible to efficiently detect ‘motion’ at the highest scale and tune the filter with it?*
   When a lot of motion is present in video, more artifacts can be expected (due to encoding and the available bit-rate), but can be removed effectively.
   The ‘highest scale’ from the multi-scale image-pyramid contains the most blurred image. This image can be useful in detecting motion, as it filters out (blurs) small details, leaving a more ‘important objects’-like representation ideally suited for detecting this kind of ‘significant motion’. Also, noise and artifacts will be less visible in the highest scale. Additionally, higher scales of an image can be stored as down-sampled versions of the input, so storage and thus processing requirements rapidly decrease. It is interesting to investigate the possibilities of determining whether a measure of the amount of motion can be derived without much extra (processing) costs and whether this measure can be used to (even locally) adapt the amount of filtering.

4. *Does a full-size approach improve image quality?*
   In an early stage of the internship it appeared that it might be beneficial to work with full-size images, instead of sub-sampling them as the scale increases (which is more usual in multi-scale processing).
   The most important reason is that this approach considerably simplifies the implementation and testing of several versions of the algorithms and improves readability (and thus maintainability) of the implementations. Additionally, as no interpolation errors occur due to interpolation of the sub-sampled data to its original resolution, it is interesting to see if this actually improves image quality.
Section 2

Introduction to multi-scale image enhancement

A multi-scale representation of an image is an ordered set of derived images intended to represent the original signal at different levels of scale. Figure 2.1 depicts such a representation. It is widely used in image and video coding and compression, enhancement of medical data (e.g. images from MR scanners), as well as many other areas of computer vision. An excellent introduction to multi-scale analysis is given in [1].

Figure 2.1: The multi-scale representation of an image can be depicted as a pyramid (Gaussian Pyramid)

One of the main advantages (in the subject of this project) is the possibility to influence components in the frequency domain, using detailed information from the spatial domain. For example, to suppress high-frequency components in areas that appear to be 'flat' (e.g. to
reduce blockyness in a clear blue sky) but enhance those components near a not-too-sharp edge (e.g. to enhance the sharpness of that edge).

To build a multi-scale representation of an image, many methods can be used. These include the Discrete Cosine Transform (DCT) used in JPEG/MPEG, Wavelet Decomposition Transform (WDT), Gaussian Pyramid (see figure 2.1) and Laplacian Pyramid. The last two are used in this project, yielding stacks of low-pass and quasi-band-pass filtered images, ideally suited for noise reduction and contrast enhancement, respectively.

2.1 Laplacian Pyramid

The aim of multi-scale image enhancement is to improve local contrast, sharpen edges and reduce noise and artifacts by modifying frequency components in the images.

This can be accomplished by decomposing the image into a set of band-pass filtered images and a DC-like residual image. When correctly built, these images can simply be summed to re-obtain the original image, or in case the band-pass images are (locally) modified, an enhanced version of the image.

A set of band-pass filtered images can be created by subtracting a low-pass filtered version of the original from the original itself, and repeating this process for subsequent bands by again filtering the low-pass version of the image. Usually a Gaussian filter is used (in this project, a filter with a separable 3 x 3 kernel using [1 2 1] as the coefficients is used to filter the images). When these images are subtracted, so called Laplacians are created.

It is impossible and not necessary to calculate images for all frequency bands. Usually only \( N = 3 \) band-pass images are calculated, the residual low-pass image is then used as the DC-like starting point for the reconstruction.

Let \( G_0 \) be the original image. Then, the Gaussians can be formulated as:

\[
G_i = \text{GaussianBlur}(G_{i-1}), \quad (1 \leq i \leq N)
\]

The band-pass filtered images are the Differences of Gaussians (DOGs), so:

\[
D_i = G_i - G_{i+1}, \quad (0 \leq i < N)
\]

Note that the above definitions imply that:

\[
G_0(x, y) = G_N(x, y) + \sum_{i=0}^{N-1} D_i(x, y)
\]

or, in a more compact notation:

\[
G_0 = G_N + \sum_{i=0}^{N-1} D_i
\]
To enhance an image (some of) the pixels of the DOGs must be modified. This is accomplished by creating a matrix $B_i$ containing the per-pixel multiplication coefficients. Coefficients less than 1 are used for noise and artifact reduction, coefficients greater than 1 are used for contrast and edge enhancement.

The enhanced version can now be calculated:

$$O = G_N + \sum_{i=0}^{N-1} B_i \cdot D_i$$  \hfill (2.1)

(where "\cdot" is the Hadamard product, or element-wise multiplication).

A graphical representation of the framework is given in figure 2.2 (for $N = 3$), figure 2.3 shows a typical sequence of Gaussians and DOGs.

![Figure 2.2: Graphical representation of multi-scale image enhancement framework](image)

![Figure 2.3: Gaussians and Differences of Gaussians (DOGs), as they are created by the algorithm in figure 2.2 (for the DOGs: 0 = grey, -128 = black, +127 = white)](image)

An efficient way of building the Laplacians is the Laplacian Pyramid implementation given
in [2], where it is used to compress image data. This implementation utilises the low-pass filtering effect of the Gaussian blur to sub-sample the image without introducing alias. Storage requirements and computational complexity are thus drastically reduced for each step up the scale (i.e. lower spatial frequencies) by a factor of 4 per step. This leads to the framework as shown in figure 2.4.

Figure 2.4: Sub-sampling version of multi-scale framework. Note the combined Low Pass Filter and Subsampler.

2.2 Contrast boosting

The multi-scale framework is introduced in the previous subsection. In this subsection the framework is extended by defining a possible set of gain-matrices $B_i$. The described algorithm is created by Harm van der Heijden [3].

2.2.1 Definition and caveats

Contrast boosting is achieved by amplifying (certain parts of) the DOGs to improve the local contrast and/or 'sharpen' edges. This means the coefficients in the gain-matrix $B$ must be greater than 1. Simply setting all coefficients to a value of e.g. 5 yields very unattractive images, see figure 2.5. A so called halo appears around James Bond's left shoulder, noise and artifacts in the hatch on the left are 'enhanced', faces appear older because wrinkles are boosted too. Though a gain of 5 is somewhat high, it can be seen that some image features are nicely enhanced, like the inner edge of the hatch.

To improve the enhancement, constraints are put on the boost-coefficients. The constraints will be discussed in the next subsections, but can be summarised as follows:
2.2. CONTRAST BOOSTING

Figure 2.5: Boosting artifacts. All three DOGs are boosted with a constant factor of 5.

- if the (per-pixel) value of a DOG is small, the corresponding coefficient must be reduced to prevent amplification of noise and artifacts;
- if the (per-pixel) value of the DOGs are large, the coefficients must be reduced to prevent halos, as the feature is already very prominent.

2.2.2 Preventing noise amplification

The first constraint can be implemented as:

\[ B_{i,\text{limit}} = \min(1, |D_i| / T_{\text{limit}}), \quad (i < N) \]

where \( T_{\text{limit}} \) is usually taken to be 8. (Note that the functions and operators used here operate element-wise, i.e. on per-pixel values!)

The matrix \( B_{i,\text{limit}} \) now contains values between 0 and 1 and is used to limit the coefficients in the matrix \( B_{i,\text{CATS}} \), defined in the next subsection.

2.2.3 Preventing halos

The second constraint is somewhat harder to achieve, because a sharp edge manifests itself in all DOGs. If pixels in the lowest-scale (i.e. high frequency) DOG indicate a sharp edge, this information should be passed on to higher scales. As the effect of an edge in a DOG spreads out in higher scales (lower frequencies have larger 'wavelengths'), the information from lower scales must also be spread out.

To accomplish this, another set of matrices is created. They will be called the Cumulative AcTivity Signals (CATS):

\[
\begin{align*}
C_0 &= \max_{k \times k}(|D_0|) \\
C_i &= \max_{k \times k}(|D_i|) + C_{i-1}, \quad (0 < i < N)
\end{align*}
\]

Here, the \( \max_{k \times k} \) operator is used to protect the edges from being boosted too much, by marking the pixels in the direct neighbourhood as 'active'. See figure 2.6 for a graphical
representation of the construction of the CATS. The width of the $\max_{k \times k}$-kernel is determined by the following factors:

1. If it is too small, halos will be suppressed near the edge, but may appear as a faint line a few pixels from the edge (see figure 3.2);
2. Higher scales need larger kernels, because information from a high-frequency edge must cover the wider range of a low-frequency edge;
3. If it is too big, it prevents features near an edge to be boosted;
4. Kernels should be kept as small as possible to limit computational complexity.

When using the down-sampling implementation of the Laplacians, item 2 is already taken care of, the kernel will then have the same size for all scales. When using the full-size approach (all matrices are stored in their full resolution, see subsection 3.2) the kernel has to be increased but can be ‘sub-sampled’ by adding zero’s, keeping the number of operations (almost) the same for all scales in case of implementation on e.g. a PC-platform. In a line-memory implementation, the number of memories will increase in this case.

The effect described in item 1 was found to occur with $3 \times 3$ kernels and disappeared with $5 \times 5$ kernels (see subsection 3.3.2), which is taken to be the default in the rest of this project. Figure 2.7 shows the CATS generated from the DOGs in figure 2.3.

The CATS basically limit the gains, so that DOGs are no longer boosted above a certain threshold:

$$B_{i,CATS} = 1 + (MaxGain - 1) \cdot \max \left( \frac{MaxCATS - C_i}{MaxCATS}, 0 \right), \ (i < N)$$
2.3. **NOISE REDUCTION**

2.3.1 **Definition and caveats**

As opposed to contrast enhancement, noise reduction is the process of suppressing frequency components in the image to reduce noise and coding artifacts like block edges, preferably in flat areas (where they are most visible). However, reduction must not be done on or near edges, as this would simply lead to blurred images, and moreover great care must be taken not to wash out relevant detail in the image. Figure 2.10 shows an example of a noise reduced version of an image (exaggerated). It can be seen that faces can look like plastic (so called 'digital make-up') and textures (like brick walls) may disappear or get blurred.
SECTION 2. INTRODUCTION TO MULTI-SCALE IMAGE ENHANCEMENT

Figure 2.9: Contrast 'enhanced' images. Settings are $T_{\text{noise}} = 8$, $\text{MaxCATS} = 64$, $\text{MaxGain} = 5$, unless otherwise stated.
2.3. NOISE REDUCTION

For noise reduction, the coefficients in the gain-matrix $B$ must be smaller than 1 in noisy areas, and equal to 1 to retain edges and other detailed parts of an image.

![Original Image](image1.jpg) ![Enhanced Image](image2.jpg)

(a) Original  (b) 'Enhanced' version, very blurred ($\text{Sharpness} = 4$)

Figure 2.10: Noise reduction artifacts

2.3.2 Implementation

Usually noise is small compared to edges and textures. The CATS matrices defined in the previous subsection can be used again to indicate the parts where noise might be visible: if the CATS is small, the DOG is also small and not too close to an edge. This yields:

$$B_{i,\text{noise}} = \min(\text{Sharpness} \times C_i, 1)$$  \hspace{1cm} (2.2)

where $\text{Sharpness}$ determines the amount of noise reduction (lower values mean more reduction, and thus more blurring). This value is usually taken between $\frac{4}{128}$ (strong reduction) and $\frac{32}{128}$ (almost no reduction).

2.3.3 Eliminating CATS

Equation (2.1) can be rewritten to:

$$O = G_N + \sum_{i=0}^{N-1} B_i \cdot D_i = \sum_{i=0}^{N-1} B_i \cdot (G_i - G_{i+1}) + G_N = \sum_{i=0}^{N} F_i \cdot G_i$$

with

$$\begin{cases} 
F_0 = B_0, \\
F_i = B_i - B_{i-1}, \quad (0 < i < N) \\
F_N = 1 - B_{N-1}
\end{cases}$$

The output image can now intuitively be seen as the weighted average of the Gaussians: select pixels from $G_N$ for strong reduction or pixels from $G_0$ to preserve an edge by appropriately setting coefficients in $F_i$. An important consequence of this is that the sum of all $F_i$ must yield a matrix containing only 1's. This restriction is satisfied by the cumulative property of the CATS matrices and the min operator in $B_{i,\text{noise}}$.

For small values in the DOGs (noise), the difference between two CATS matrices is approximately equal to $|D_i|$. This fact, and the fact that the sum of the matrices must be one,
can be used to simplify the matrices $F_i$ to:

$$
egin{align*}
F_0 &= B_0 = \min(S \ast C_0, 1) = \min(S \ast |D_0|, 1), \\
F_i &= B_i - B_{i-1} = \min(S \ast C_i, 1) - \min(S \ast C_{i-1}, 1) \approx \min(S \ast |D_i|, L(i)), \quad (0 < i < N) \\
F_N &= 1 - B_{N-1} \approx L(N)
\end{align*}
$$

where $S$ is the abbreviation of *Sharpness* and $L(i) = 1 - \sum_{j=0}^{i-1} F_j$, i.e. the left-over part of the weight that can be distributed among the $F$'s, keeping the total at 1.

It is now clear that the gain matrices $F_i$ can be calculated using only the absolute values of the DOGs.

A typical set of Gaussians and gains is depicted in figure 2.11, showing that edges are taken from the sharper versions of the images, while the noisy air is mostly taken from the most blurred Gaussian. Some example images enhanced using this version of the algorithm are given in figure 2.12.
Figure 2.11: Gaussians and corresponding gains, \textit{Sharpness} = 8, light grey = gain of 1
Figure 2.12: Noise reduction examples, originals (left) and enhanced (right), $\textit{Sharpness} = 8$. Please note that differences will not be clear (and even misleading) in a printed version of this document.
Section 3

Enhancements

In the previous section the foundation for a multi-scale approach to image enhancement is built. As most of the intermediate steps to calculate the output can be seen as images, a tool to view these images simultaneously with the output is very helpful for making improvements to the algorithms. This tool is described in the first subsection.

To facilitate in readable and maintainable code, images are processed in their full resolution, in contrast to the sub-sampling approach described in subsection 2.1. In subsection 3.2, the implications of using full-size images are discussed.

Subsection 3.3 deals with the process of defining an integrated algorithm to simultaneously boost contrast and reduce artifacts and noise.

In the last subsection the algorithm is extended from still images to video streams, to automatically adapt the filtering to the 'quality' of the content.

3.1 Graphical User Interface (GUI)

During the internship a GUI was developed giving insight into the internal behaviour of the algorithms, which led to improvements in the determination of the gain-matrices, enabling the integration of noise reduction and contrast boosting in one algorithm.

The GUI evolved from a simple view of the matrices into a powerful tool being able to magnify parts of the image, manipulate parameters of the algorithm, save screen dumps of individual images, view histograms, etc. Almost all operations work even when processing a live video stream. The application is built up of modules, which enables it to be used in a wide variety of processing frameworks. Currently, the application is available as plug-ins for AviSynth [4] and VideoPump (an internal research tool for the Video Processing group at Philips NatLab) and of course as a stand-alone version.

Besides the visual feedback of the matrices, it is important to be able to easily make
changes to the tested algorithms. Therefore, a more Matlab-like way of implementing the algorithms in code is desirable, while maintaining the high speed of a 'conventional' programming language.

This calls for a RAD (Rapid Application Development) programming language. Because of the familiarity of the author with Borland Delphi 7, this language was chosen as the implementation environment. It has the advantages that a GUI is easily created and code is comprehensible, even for people that have never seen the language before.

As most algorithms are mathematically described by operations on entire images, this approach is also used in the Delphi code. This does give some overhead in terms of computational complexity, but MMX optimisations of some (elementary) image-operations make the code run real-time for most experiments.

A typical view of the GUI is shown in figure 3.1.

![Figure 3.1: Typical view of the GUI. At the top are controls to adjust filter parameters. In the left column, from top to bottom, GUI-controls, the output and the Gaussians can be seen. In the centre, the full-resolution version of a selected image is displayed and to its right a 'magnifying glass' and histogram. Below the big image are the DOGs, CATS and Gains corresponding to the Gaussians. To the right of them are the 1D GainTable, the 2D GainTable and a 2D lookup table containing DOG*Gain.](image-url)
3.2. FULL-SIZE APPROACH

Some things to note about its usage:

- When AUTO SELECT is enabled, the big image is automatically updated to reflect the image that is under the mouse-cursor. Use the A-key to toggle AUTO SELECT. AUTO SELECT is automatically disabled when an image is clicked. Use I and O to select the Input and Output image respectively. Note that these shortcuts only work in stand-alone mode.

- Press and hold the Alt-key while clicking on the big image to save that image to a file.

- In stand-alone mode the pull-down menu contains images from the folder “..\Images\”.

- It is possible to view an even larger image of the big image by clicking it. In this view, the zoom factor and aspect ratio can be changed by clicking with the right mouse button. Also, a part of the image can be saved by dragging with the mouse and then Alt-clicking the image. Other images can be selected from the GUI, keeping the selection-rectangle in place, so saving the same part of a sequence of images is possible.

- The histogram and gain tables can also be displayed as a big image by clicking.

- Use the INSTANT PREVIEW option to update the GUI with live images from the video stream.

- The histogram displays the peak value, and where that peak occurs. The image is automatically scaled to fit the largest peak. If this peak is too big compared to the rest of the histogram, the second largest peak is determined and used to scale the image. This peak will then also be displayed next to the image. The second-peak mode is useful if one value (like black) would otherwise dominate the picture and cause the other values to appear as ‘bars’ of 1 pixel high.

3.2 Full-size approach

Because the DOGs are calculated by subtracting one Gaussian from another one, these Gaussians must have the same dimensions. In the efficient implementation mentioned in subsection 2.1, the images are down-sampled and must first be up-sampled again. This is usually accomplished by simple linear up-sampling. Advantages of this approach are the lower computational complexity and reduced memory usage and bandwidth.

However, it is interesting to investigate the possibilities of working with full-size images, because of the following reasons:

- The processing required for interpolation is eliminated;

- There are no interpolation errors, which might be beneficial for image quality (see subsection 4.1);
• Any part of the algorithm can directly use an image from any other part of the image without having to worry about dimensions, yielding good readability and maintainability of the code.

The last item is the main reason the full-size approach was chosen.

Working with full-size images implies working with bigger kernels for higher scales. It is possible however, to use a sub-sampling technique for these kernels, by inserting zeros in the kernel. For example, the 1D separated kernel to calculate the first Gaussian might be:

\[
\begin{bmatrix}
1 & 2 & 1
\end{bmatrix}
\]

while the kernel for the next Gaussian will then look like:

\[
\begin{bmatrix}
1 & 0 & 2 & 0 & 1
\end{bmatrix}
\]

This way, the number of operations per pixel will remain the same for every scale, as the zeros in the kernel can be skipped in the calculations.

### 3.3 Combining boosting and reduction

Noise reduction is the process of reducing frequency-components in the image, whereas contrast boosting tries to amplify them. Thus, to prevent cancellation of both effects, care must be taken when combining these two.

The problem is already partly solved by the used algorithms, because the noise reduction-step only tries to modify very low detail, and contrast boosting only tries to modify medium detail.

Therefore, a simple way to combine the algorithms would be to put them in a cascade. The result is not satisfactory though, because there will be a clear distinction between the parts where noise is removed, and parts where the contrast is boosted. Especially with faces, this effect looks like some sort of thresholding. This calls for an integrated solution, where both reduction and boosting are carried out by the same algorithm, ensuring a smooth transition from e.g. a noise-reduced background to a contrast-enhanced edge.

In this subsection, the foundations of the previous section are analysed and improved. First, noise reduction is given a closer look, then the CATS matrices are discussed and lastly the gained insights are used to present an integrated algorithm.

#### 3.3.1 Noise reduction

In subsection 2.3 the coefficients of the gain are calculated using a linear relation (equation (2.2)): $\text{Sharpness} \cdot C_i$. Several other relations were tried, like $\text{Sharpness} \cdot C_i^2$ and
3.3. COMBINING BOOSTING AND REDUCTION

\textit{Sharpness} \* \sqrt{C_i}. Quadratic and higher order relations yield slightly better results, but can introduce other problems like contouring. The reason is the very limited number of values where the formula is used, as for $C_i \gtrsim 8$ the gain is already saturated to 1. So when using e.g. a 4\textsuperscript{th} order version, the gain-increase between $C_i = 7$ and $C_i = 8$ is very large and will have the effect of thresholding: it will seem like the noise reduction is turned on and off for small parts in the image. Hence, care must be taken to have a smooth transition between noise-reduction and pass-through (or even boosting).

Although noise reduction can be achieved very efficiently with just the Gaussians and their absolute differences (see subsection 2.3.3), this approach cannot be used for contrast boosting: there is no enhanced version to 'choose with the weighted average'. Simply taking gains greater than 1 will not work because the amount of enhancement would then depend on the grey-values of the pixels.

Therefore the version with CATS and DOGs will be used in the rest of this document.

3.3.2 CATS revisited

The CATS matrices are used to limit the amount of contrast boosting in case of strong edges, or to protect otherwise detailed areas. The same matrices can be used to detect parts in the image without much detail, i.e. areas where noise will be more visible. As the gains are directly determined from CATS (and DOGs), this signal will have a great influence on the output image.

Extended halo effect

A picture of the CATS, as in figure 2.7, shows that it becomes white for very strong edges, black for areas where no significant detail is found and greyish for areas that contain a moderate amount of detail. Intuitively, one could state that the gains could then be derived from the CATS by having a very low gain (near zero, for noise reduction) for very low CATS, a high gain for the greyish areas (low contrast that can be boosted) and unity for high CATS (to prevent halos).

This leads to images as in figure 3.2. A faint line can be seen at a distance from an edge, as if the halo is being suppressed for just a few pixels. This is exactly what happens, but the line wouldn't be visible if it wouldn't have been amplified. The amplification is caused by the gradual decrease of the CATS around an edge: it is white at the edge and the $k \times k$ area around it, but fades to dark in a few pixels (instead of with a step-like function), traversing the greyish range. This greyish range yields the high gains responsible for the 'extended halo' (note the white lines surrounding edges in the gain-image).

Methods tried to reduce this effect include:

- Using other kernel shapes for the $\text{max}_{k \times k}$-function: lines that appear as a square around
20

SECTION 3. ENHANCEMENTS

Figure 3.2: ‘Extended halo’ effect. Note the faint line above the head in the left image. It is caused by the high gain as can be seen in the right image. A $5 \times 5$ kernel can solve this problem.

- Using larger kernels: the impact of a strong edge diminishes with the distance from it, so the effect of a high gain will be less;

- Using the CATS of a higher scale to reduce the gain by taking the minimum of the gain calculated using the ‘current’ CATS and the gain calculated using the next level CATS: this works for all scales but the highest, so it is not a good solution;

- Algorithms that try to find and eliminate the thin white lines in the gain-images, and/or make the transition from dark to light instantaneous: works, but is computationally intensive;

- Include the DOG in the calculation of the maximum gain: works, and was actually already used in the algorithm, though with a slightly other purpose and ‘weight’.

It appeared that using a combination of CATS and DOGs gives the best results, but also the size of the max-kernel can make a difference in some cases. Furthermore, the value of MaxCATS should not be taken too low for a certain MaxGain.

Kernel shapes and sizes

A 9-tap ‘diamond-shape’ kernel (see figure 3.3) was tried (this shape is only possible for higher scales) and gave fair results in most cases, slightly better than a ‘normal’ $3 \times 3$ kernel. But even then scenes showed up where the ’extended halo’ effect was noticeable. Increasing the size of the square kernel to $5 \times 5$ helped to solve the problem completely. As this effect only occurs at the highest scale, this kernel size is only used for the highest scale. For lower scales, a $3 \times 3$ kernel is used.
3.3. COMBINING BOOSTING AND REDUCTION

Because of the low-pass filtering effect of the Gaussians, the size of the kernel needs to be expanded for every scale when using full-sized images. This would require a $17 \times 17$ kernel to calculate the third CATS, which is hardly feasible for real-time use. Fortunately, it is possible to also 'sub-sample' the max-kernel, just like the Gaussian kernel.

This works very good for sharp edges, as the value of the CATS will vary between 0 and 255, but only 0 to $\text{MaxCATS}$ is used, so all sharp edges appear as saturated white. For medium-detail areas, an effect as illustrated by figure 3.4 occurs. This effect can be reduced by e.g. (Gaussian) blurring the CATS themselves before using the $\max_{k\times k}$ operator, but this did not yield noticeable improvements to the final image output.

![maxkxk kernel sub-sampling artifacts](image)

Figure 3.4: $\max_{k\times k}$ kernel sub-sampling artifacts. Note the grid-like patterns.

3.3.3 Integrated boosting and reduction

To successfully integrate both goals the effect of gain matrix $B_{i, \text{boost}}$ from subsection 2.2.4 must be studied, as the key of the algorithm lies in improving that matrix.

The gain matrix $B_{i, \text{boost}}$ produces a multiplication factor based on both CATS and DOGs.
To calculate the value of an element in $B_{i,boost}$, one could take a two-dimensional lookup table. This lookup table can be visualised, by taking the colour black for a gain of zero and white for the maximum gain and some shade of grey for the gain of 1, putting CATS on the horizontal axis and DOGs on the vertical axis.

Figure 3.5 shows this visualisation, using white for a $MaxGain$ of 8 and with $T_{noise} = 16$ and $MaxCATS = 64$. Note that the figure is symmetrical around the horizontal axis, because the absolute value of the DOGs is taken. Note also that the parts where the DOG is larger than the CATS (i.e. the upper-left and lower-left parts of the image) are, by definition of the CATS, 'unused'.

The figure clearly shows the vertical gradient preventing small DOGs to be boosted too much and the horizontal gradient for the protection of very sharp edges.

When looking at the meaning of CATS and DOGs in images, the following can be concluded:

- The area near the origin (low values for both CATS and DOGs) denotes parts in the image that contain no significant detail, like a clear blue sky or large smooth surface;
- When DOGs and/or CATS become somewhat larger, this denotes parts where medium detail occurs;
- When CATS become really large, this denotes a sharp edge.

This insight leads to a new lookup table, shown in figure 3.6.

The CATS signal still protects sharp edges, but small DOGs can now still be boosted if the CATS signal signifies enough detail in the area. Noise is now not just passed-through, but actively suppressed if both CATS and DOGs are small.

The new table is formed by taking the minimum of the already known $B_{i,CATS}$ and the new $B_{i,circle}$:

$$B_{i,integrated} = \min (B_{i,CATS}, B_{i,circle})$$
3.4 CONTENT ADAPTIVE FILTERING

Figure 3.6: Improved lookup table

With:

\[ B_i,circle = \text{GainTable} \left( \sqrt{|D_i|^2 / \text{CircleShape} + C_i^2} \right) \]

and:

\[ \text{GainTable}(x) = \begin{cases} 
    x \ast \frac{(\text{MaxGain} - \text{Sharpness})}{\text{CircleRadius}} + \text{Sharpness}, & x < \text{CircleRadius} \\
    \text{MaxGain}, & x \geq \text{CircleRadius}
\end{cases} \]

The filtering can thus be tuned by several new parameters:

- **Sharpness** determines the amount by which the small DOGs are suppressed. This value is typically taken between 0.25 and 0.5.
- **CircleRadius** sets the width of the ramp from Sharpness to MaxGain. If this value is large, bigger DOGs will still be suppressed. It is typically set between 24 and 64.
- **CircleShape** can be used to deform the circle into an ellipse. This way, the ratio between influences of CATS and DOGs can be changed. Usually set between 0.1 and 1.0, but mostly just 1.0.
- The meaning of **MaxGain** remains the same, but it should be noted that changing the value of this parameter not only controls the maximum gain, but also affects the amount of noise reduction as it influences the slope of the ramp.

Examples of the effects of changing these parameters are given in the Benchmarks section.

3.4 Content adaptive filtering

The integrated algorithm from the previous subsection is controlled by several variables like the amount of reduction and maximum gain. The actual settings will depend on the quality of the material, which might in fact be different from frame to frame.
Video streams, especially those with a fixed bit-rate like in digital television broadcasts, will suffer from scenes where lots of changes occur between frames (see figure). The shortage of bandwidth will result in artifacts like blocking and ringing, but could effectively be suppressed by temporarily (and even locally) increasing the amount of artifact/noise reduction. Figure ?? illustrates the occurrence of blocking artifacts in the two men running down the hill and the dust-cloud behind them.

Figure 3.7: Blocking artifacts because of large motion. Note the block-edges on the two men running down the hill, and in the dust-cloud following them.

3.4.1 High Scale Motion Detection (HSMD)

Full-frame metric

To detect changes between frames, the absolute difference between the most blurred Gaussian of the current and the previous frame is taken. Changes due to noise, blocking, ringing, small movements, etc. are blurred enough to have no noticeable influence on this 'motion image' for most cases.

In mathematical notation:

\[ M_j = |G_{N,j} - G_{N,j-1}| \]

with \( j \) the frame number in the video sequence.

Matrix \( M \) contains 'amount of change'-information per pixel, which can be converted into a single global parameter:

\[ m_j = \frac{\sum_{x,y} M_j(x,y)}{W \times H} \]

with \( W \) and \( H \) being the width and height of the frames.
3.4. CONTENT ADAPTIVE FILTERING

This parameter is an indication of the amount of filtering required, and will control the settings of the integrated algorithm of section 3.3.3.

Lookup table selection

As on-the-fly calculation of the 2D lookup tables is not feasible for real-time use, a pre-determined set of lookup tables will be constructed. The first table will be used in case of no 'motion' and subsequent tables with increasing 'amount of change'. One of those lookup tables can be selected per frame, according to the value of \( m_j \), e.g. using:

\[
T_j = \min \left( \text{MotionSensitivity} \times m_j, N_{\text{tables}} - 1 \right)
\]

with \( N_{\text{tables}} \) the number of pre-calculated tables, so that \( 0 \leq T_j < N_{\text{tables}} \). In the current implementation \( N_{\text{tables}} = 11 \). The parameter \( \text{MotionSensitivity} \) is used to tune the aggressiveness of the filtering and should be set to between 0.5 and 1.0 in this setup.

As explained in the introduction of this subsection, the (MPEG) codec will introduce artifacts when large frame differences occur due to the limited bit-rate. It will take some time (frames) to 'recover' from these artifacts, which is incorporated into the final selection of the table:

\[
\begin{cases}
I_j = T_j, & (T_j \geq T_{j-1}) \\
I_j = T_{j-1} - \min(T_{j-1} - T_j, \text{MaxChange}), & (T_j < T_{j-1})
\end{cases}
\]

This approach ensures an immediate fall-back to strong reduction, with a slow recovery (controlled by \( \text{MaxChange} \)) to the sharp version.

Lookup table creation

A certain table will now be selected, based on the amount of motion/change between frames. These tables must have different properties that are determined by the values of the parameters as described in subsection 3.3.3.

The following settings can be changed for increasing table-indexes (i.e. increasingly strong noise/artifact reduction):

- decrease \( \text{Sharpness} \), e.g. from 16 to 0;
- increase \( \text{CircleRadius} \), e.g. from 32 to 56;
- decrease \( \text{MaxGain} \), e.g. from 4 to 1.5.

Especially changes in \( \text{MaxGain} \) are very visible, which calls for an independent, non-linear tuning of the parameters. In the first few tables, only \( \text{Sharpness} \) and \( \text{CircleRadius} \) are affected. Then, for very strong movement, also the \( \text{MaxGain} \) is reduced.
The changes in parameters are specified in lookup-tables, like:

\[
\begin{align*}
\text{SharpnessSteps} &= [1.0, 0.9, 0.8, 0.6, 0.4, 0.2, 0.0, 0.0, 0.0, 0.0] \\
\text{CircleSizes} &= [1.0, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.7] \\
\text{GainSteps} &= [1.0, 1.0, 1.0, 0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.4]
\end{align*}
\]

Now for every \( t \in [0, N_{\text{tables}}] \) the parameters change according to:

\[
\begin{align*}
\text{Sharpness}_t &= \text{Sharpness} \times \text{SharpnessSteps}[t] \\
\text{MaxGain}_t &= \text{MaxGain} \times \text{GainSteps}[t] + 1 \times (1 - \text{GainSteps}[t]) \\
\text{CircleRadius}_t &= \text{CircleRadius} \times \text{CircleSizes}[t]
\end{align*}
\]

The calculation of the lookup-tables is then simply performed using the ‘\( t \)-versions’ of the parameters.

Note that these formulas are designed for low-quality input (e.g. 800 kbit/s MPEG-1, Standard Definition). For high-quality input (e.g. 8000 kbit/s MPEG-2, Standard Definition) a smaller \textit{MotionSensitivity} can be used, so only the first few tables will be selected.

**Block-based metric**

The full-frame method works well for most videos, but for largely stationary scenes where a lot of motion occurs, the \textit{MaxGain} is also decreased. This leads to the scene appearing to get out of focus, and after the movement get back into focus again; a very annoying effect. This calls for a local variant of the HSMD algorithm.

The change-matrix \( M_j \) provides ‘per pixel’ information about the motion. Attempts to modify the algorithm to a per-pixel version did not yield satisfactory results though: the slow-decay-property is computationally complex and the appearance of block-edges depends on areas bigger than a single pixel. A better approach is to divide the image in several blocks (64 in the current implementation) and determine the table index to use per block.

This also enables the use of a larger motion sensitivity \textit{MotionSensitivity}, enabling even stronger filtering for blocky parts in a frame.

**3.4.2 Other metrics**

Other metrics can also be included in the algorithm, for example to help to choose which lookup-table to use. For this project, the Beacon Block Grid Detector [5] was used, which includes a Blockyness Impairment Metric [6].

It appeared that the per-frame values of this metric were not reliable enough to decide on picture quality. An IIR filtered version (a running average) did provide good results, and enables to distinguish between e.g. high-quality DVD content and low-quality DivX content.
3.4. CONTENT ADAPTIVE FILTERING

Using $BIM_{avg} = BIM_{avg} + 0.01 \times (BIM_{frame} - BIM_{avg})$, the value follows the (constantly changing) quality of the input fast enough, while still providing a reliable 'quality signal'. Experiments with this indicator have shown that values around 1.20 indicate good quality (DVD) content, values around 1.40 indicate medium quality (blockyness starts to be visible) and values around 1.70 and higher denote very blocky content where almost no boosting can be performed and just strong noise reduction is advised.

In figure 3.10, a high-quality DVD source (MPEG-2, 9801 kbit/s, 720x352) is compared to a re-encoded version of the source (MPEG-1, 1000 kbit/s, 720x352). Pictured are the 'real-time' (unfiltered) BIM values of both streams, and their averaged versions. From this figure it is clear that the filtered version can indeed be an indication of the input-quality. The unfiltered version is very 'noisy' and will make the video have a so called 'breathing effect': details will continuously appear to get in and out of focus.

![Figure 3.8: Real-time BIM values versus filtered versions](image)

In figure 3.9, the filtered versions for different bitrates using the MPEG-1 codec are plotted. In figure 3.10, this is done for different versions using the DivX codec. The filtered BIMs in this last figure do not give a good indication of the input quality, which can be explained by the fact that these algorithms introduce other kinds of artifacts than blocking, and are as such not measured by the BIM algorithm.
SECTION 3. ENHANCEMENTS

Figure 3.9: Filtered BIM for MPEG-1 streams

Figure 3.10: Filtered BIM for DivX streams
Section 4

Evaluation

This section starts with an evaluation of the advantages and disadvantages of using full-resolution images throughout the implementation (as is used in the Graphical User Interface implementation). Next, the effects of changing the parameters of the integrated algorithm are explained and illustrated, after which some typical examples are shown for different kinds of input quality. The section concludes with a discussion on the efficiency of the algorithm.

4.1 Full-size versus Sub-sampled

The artifacts of sub-sampling followed by linear interpolation are shown in figure 4.1. The image is obtained by up-converting the most blurred Gaussian ($N = 3$) to the original resolution. Clearly, the pyramid-like interpolation is visible. This effect is even more prominent in video, as the 'grid' has a fixed offset just like the encoding blocks in an MPEG-stream.

However, these effects are nearly not noticeable when viewing movies with an algorithm as described in Section 2. This is because the interpolated version of the most blurred Gaussian is only used for very flat areas, where this effect is not visible. In other words: for increasingly sharper features an increasingly sharper Gaussian is used.

Sub-sampling has the advantages that it consumes less memory and generally requires less processing power. The full-size approach on the other hand facilitates algorithm research, saves some processing by eliminating the interpolation step and doesn’t introduce interpolation artifacts. A practical full-size implementation does suffer from some sub-sampling artifacts though, because of the sub-sampled Gaussian- and $\max_{k\times k}$-kernels. Therefore, sub-sampling is preferred in real-time applications, and full-size may be more attractive for research projects.
4.2 Effects of parameter changes

In the examples below, the face of James Bond was taken to demonstrate the effects of all parameters. In the case of changes to noise reduction parameters, two settings for the MaxGain are used: 1.00 and 3.50. These values represent no boosting (just noise reduction) and moderate boosting.

4.2.1 Sharpness

The Sharpness is the vertical starting point of the 1D GainTable from subsection 3.3.3 (not to be confused with the 2D lookup tables $B_{i, integrated}$) and as such controls how much a certain DOG can be reduced. E.g., if this value is taken as 0.5 (16 in the GUI (0.5 * 32)), then the smallest DOG/CATS will be multiplied by 0.5.

This value can be set to 0.5 or higher (but ≤ 1) for high-quality content, and to 0.0 for low-quality content.

4.2.2 CircleRadius

The CircleRadius controls the radius of the circle extending from the origin into the 'MaxCATS-gradient'. A smaller CircleRadius means that fewer details will be marked as artifacts. It therefore influences both the amount of noise reduction and (inversely) the amount of boosting. So for low-quality content the noise reduction should be increased, but this automatically limits the amount of boosting that is possible.

High-quality content can use a setting of 24, whereas low-quality content is better of with 56, with a corresponding MaxGain of 2.50.
4.2. EFFECTS OF PARAMETER CHANGES

Figure 4.2: Effect of changing Sharpness in absence of boosting (CircleRadius = 32, MaxGain = 1.00, MaxCATS = 64)
SECTION 4. EVALUATION

Figure 4.3: Effect of changing Sharpness with constant boosting (CircleRadius = 32, MaxGain = 3.50, MaxCATS = 64)
4.2. EFFECTS OF PARAMETER CHANGES

Figure 4.4: Effect of changing CircleRadius in absence of boosting (Sharpness = 0, MaxGain = 1.00, MaxCATS = 64)
SECTION 4. EVALUATION

(a) Original

(b) Very large radius: much blurring, CircleRadius = 64

(c) Large radius: recommended for low-quality material, CircleRadius = 56

(d) Default radius: medium quality content, CircleRadius = 32

(e) Small radius: recommended for high-quality content, CircleRadius = 24

(f) Very small radius: not recommended for boosting, CircleRadius = 8

Figure 4.5: Effect of changing CircleRadius with moderate boosting (Sharpness = 0, MaxGain = 3.50, MaxCATS = 64)
4.2.3 CircleShape

The effect of this parameter is minimal, and only works for values <0.10. When used with such a low value, mostly noise will be amplified. It is included here for 'historical' reasons, and should not be used. The value is always 1.00 in this report.

4.2.4 MaxGain

Changes in MaxGain affect not only the maximum gain values, but also the amount of noise reduction (as it determines the slope of the noise reduction 'crater'). When set to 1.00, the algorithm only performs noise reduction, for higher values boosting will occur too. Please note that the maximum gain that will actually be reached is lower than the value set by MaxGain, because this would only occur for CATS = 0 and DOG ≥ CircleRadius; something that can never happen because (by definition) DOG ≤ CATS.

See also the images in figure 2.9.

4.2.5 MaxCATS

The MaxCATS setting determines how sharp an edge should be before it gets protected. Setting this value too high will introduce halos or other unwanted artifacts like clipping, setting it too low will prevent moderate detail to be boosted. The value also depends on the value of MaxGain: higher gains have a higher risk of producing halos. For most cases a value of 64 can be used.

See also the images in figure 2.9.

4.2.6 MotionSensitivity

MotionSensitivity controls how large the motion should be to switch to the next gain lookup-table. This value is usually set to 0.5, but can be made larger to have a stronger reaction to motion.

4.3 Typical examples

To give an impression of the effects of applying the filter to different kinds of input, a set of example images is presented in the following figures. The images are processed with reasonable settings for low-quality source material (e.g. 800 kbit/s DivX, Standard Definition) and high-quality material (e.g. 8000 kbit/s MPEG-2, Standard Definition). Please note that the differences may be too small to notice on the printed version of this report, and maybe even
SECTION 4. EVALUATION

Figure 4.6: Effect of changing MaxGain (Sharpness = 16, CircleRadius = 32, MaxCATS = 64)

(a) Original
(b) No boosting: only noise/artifact reduction, MaxGain = 1.00
(c) Almost no boosting: useful to 'compensate' for noise reduction, MaxGain = 2.00
(d) Moderate boosting, MaxGain = 3.00
(e) Substantial boosting, MaxGain = 4.00
(f) Excessive boosting: not recommended, MaxGain = 8.00
4.3. TYPICAL EXAMPLES

(a) Original

(b) Very low value: almost no boosting will occur, \(\text{MaxCATS} = 32\)

(c) Default value: recommended, \(\text{MaxCATS} = 64\)

(d) No CATS protection: halo’s and saturation will occur, \(\text{MaxCATS} = 255\)

Figure 4.7: Effect of changing \(\text{MaxCATS}\) (\(\text{Sharpness} = 16\), \(\text{CircleRadius} = 32\), \(\text{MaxGain} = 3.50\))
when viewing on a PC monitor. The settings chosen here are tuned for display on a bright High Definition LCD TV.

The high-quality settings are: \textit{Sharpness} = 16, \textit{CircleRadius} = 32, \textit{MaxCATS} = 64, \textit{MaxGain} = 3.50.

The low-quality settings are: \textit{Sharpness} = 0, \textit{CircleRadius} = 56, \textit{MaxCATS} = 64, \textit{MaxGain} = 2.50.
Figure 4.8: Typical example images
4.4 Efficiency of integrated algorithm

The current implementation uses 18 full resolution buffers: 4 Gaussians, 3 scratch-buffers to calculate Gaussians, 3 DOGs, 3 CATS, 3 Gains, 1 Output and 1 Motion buffer. This requires a significant amount of memory, and a large memory bandwidth, which makes this approach impractical for real-time hardware implementation.

The algorithm could however be transformed in a streaming line-memory version, where the calculation of the CATS will most likely be the largest factor because of the $5 \times 5$ maximum. Because of the incremental nature of the algorithm, and because most per-pixel values are used only once and can thus be calculated on-the-fly (like gains), the needed line-memories will be a fraction of the buffers used in this project.

To calculate the motion-buffer a whole frame has to be stored, but as this concerns the highest scale buffers this buffer only needs to be $(1/4)^3 = 1/64$ of the size of a full resolution buffer.

Considering the sub-sampled version of the algorithm, the main operations that have to be performed are the calculation of the Gaussians, the interpolation for the creation of the DOGs, construction of the CATS, determining the gain to use and finally multiplying that with the DOG and adding to the output.

An interesting side-effect of the algorithm is that especially the Gaussians and motion-buffer can be a useful contribution to other algorithms (e.g. a de-interlacer).
Section 5

Conclusions

The general conclusion of the project is that video enhancement using the multi-scale approach has promising results: it is possible to have combined artifact reduction and contrast boosting in one algorithm, for both low and high quality video. Some parts of the algorithm (notably the Gaussians and motion-buffer) can be useful for other algorithms too.

Key parameters for picture quality (to tune the amount of artifact reduction and/or contrast boosting) are determined, and a Graphical User Interface is provided to tune these settings while processing a 'live' video stream. Also, 'High Scale Motion Detection' was developed, assisting in the automatic tuning of filter settings. Other metrics (like a block grid detector) can be included as well, and are needed for automatic adjustment to different quality of input material.

5.1 Answers to the Research Questions

5.1.1 Can both noise reduction and contrast boosting be combined in one algorithm?

Yes. The integration of both algorithms makes it possible to smoothly shift from noise reduction to contrast boosting and anything in between, without thresholding artifacts that would occur when cascading the two.

The integrated version works on both high quality and low quality content, using a suitable settings. Of course, the algorithm can't do magic: if artifacts are too prominent, the distinction between real detail and e.g. an unwanted block edge can't be made, so only very moderate boosting can be expected then.

Relevant parameters concerning image quality have been derived and sensible values for different kinds of input are determined.
Problematic areas are the ageing/blurring of human faces and insufficient removal of ringing artifacts.

5.1.2 Can external indicators help to improve the output?

Yes. External indicators are a necessity to automatically tune the filter parameters according to the quality of the input. Currently, the High Scale Motion Detection is implemented, and tests indicate that also a Blockyness Impairment Metric can be used to distinguish between high (e.g. DVD) and low (e.g. DivX) quality content.

5.1.3 Is it possible to efficiently detect 'motion' at the highest scale and tune the filter with it?

Yes. At almost no extra cost a reliable signal can be created that signifies largely changing parts in the image. The block-based version of the signal is used in the algorithm to tune reduction and boosting parameters according to the expected amount of artifacts, which led to very satisfactory results.

5.1.4 Does a full-size approach improve image quality?

Not noticeably. The benefit of processing images in their full resolution, instead of saving sub-sampled versions of the Gaussians, is mostly the ease of use in the development of the algorithms. Therefore, no thorough tests on image quality have been performed, but both approaches seem to provide good picture quality.

Especially the development of the Graphical User Interface has led to a much better insight into the multi-scale approach, and helped to find suitable values for the parameters.

5.2 Discussion

Although no perception tests were performed, most people judged the sequences processed by the integrated algorithm to be 'better' than the original. With human faces and low-quality settings however, noise reduction is often immediately noticed and unwanted.

In the implementation as described in this report, the algorithm is fairly expensive in terms of memory-usage and computational complexity, but it is possible to create a line-memory implementation using the sub-sampled version which can be affordable. Also, many other algorithms could benefit from the availability of the multi-scale signals. For example, a de-interlacer can be informed that there is significant motion in a certain area so it can take appropriate action.
5.3. SUGGESTIONS FOR FUTURE WORK

5.3 Suggestions for future work

5.3.1 Metrics

Other metrics (like analog noise or ringing detectors) can be added to the algorithm, to facilitate the distinction between high and low quality content. Also skin-tone detection could be useful to decrease the amount of noise reduction for human faces, although this will probably introduce other problems.

One might think of using a block-grid detector to easily suppress e.g. the high-frequency block edges in the DOGs, but this is not as simple as it may seem, because for video streams the block-grid is severely distorted because of displacements by motion-vectors.

5.3.2 Colour

The colour (UV) data also contains artifacts, which can be processed as well. This could be done by separate processing, or by using the gains obtained from the luminance filtering.

Also, colours are desaturated when increasing the luminance, and vice versa. To prevent this, the gains from the luminance-signal, or something like \( U_{\text{out}} = U_{\text{in}} \cdot Y_{\text{out}} / Y_{\text{in}} \) could be used.

If the colour data is not processed, the greater bit-precision achieved when creating the Gaussians can be used to reduce contouring artifacts. The 'extra' bits could be used to 'dither' UV-data, to get a higher precision than the current 8-bit Y-signal. Especially gradients (e.g. a clear blue sky) could benefit from this.

5.3.3 Performance

In the algorithm as described in subsection 3.3.3 first a gain is determined using a lookup table. Then, the gain is multiplied with the DOG. As one of the 'inputs' of the lookup table is the DOG itself, the multiplication can be eliminated by including it in the table.

The integrated algorithm can be changed to a line-memory version for stream processing on specialised hardware. Another possibility is to implement it in Pixel Shaders on modern graphics cards. These shaders are perfectly suited for the scaling, interpolation and multiplication and addition steps used in multi-scale. The downside of this is that intellectual property is currently hard to protect, because the byte-code that drives the shaders can easily be extracted and disassembled, thus exposing the ideas behind the algorithm. This is not a problem for the algorithms in this report (as it is publicly published), but might be a concern if significant research is put in improved versions.
5.3.4 Training

The gain lookup table is currently defined by manually choosing suitable values for e.g. CircleRadius and MaxGain and the ‘shape’ of the table is also hand crafted. It may be very interesting to train the table from actual content. This could be done as follows:

- Take high quality video
- Re-encode this video to a low(er) bit-rate: version A
- Boost the original (e.g. using PixelPlus): version B
- Train the lookup table(s) using A and B

One could train separate tables for every scale, and even use multiple tables per scale using the HSMD algorithm.

5.3.5 Subjective Assessment

No perception tests were performed to tune the filter parameters during this internship, but a proper subjective assessment would be a logical next step if the results are going to be used for other projects.
References


