BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

PHOTOGRAPHIC DETAIL ENHANCEMENT METHODS

METODY ZVÝRAZŇUJÍCÍ DETAILY VE FOTOGRAFII

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR **Bc. TOMÁŠ HUDZIEC**
AUTOR PRÁCE

SUPERVISOR **Doc. Ing. MARTIN ČADÍK, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2018/2019

Zadání diplomové práce

Student: **Hudziec Tomáš, Bc.**
Program: Informační technologie Obor: Počítačová grafika a multimédia
Název: **Metody zvýrazňující detaily ve fotografii**
Photographic Detail Enhancement Methods
Kategorie: Zpracování obrazu

Zadání:

1. Seznamte se s problematikou metod pro zvýraznění detailu v obraze.
2. Vyberte a popište metody vhodné pro implementaci.
3. Do již existujícího systému implementujte alespoň tři zvolené metody.
4. S metodami experimentujte, posuďte jejich vlastnosti při převodu videosekvencí a LDR i HDR obrazů, implementované algoritmy porovnejte a diskutujte možnosti budoucího vývoje.
5. Dosažené výsledky prezentujte formou videa, plakátu, článku, apod.

Literatura:

- Dle pokynů vedoucího.

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 3 zadání.

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Čadík Martin, doc. Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 22. května 2019
Datum schválení: 1. listopadu 2018

Abstract

This thesis studies several methods for enhancing details in digital photographs. Methods' algorithms are described and implemented to existing system using C++ and OpenCV. Methods are then compared in terms of the time and memory complexity and their results are evaluated using users' questionnaire. Work overally gives overview of present photographic detail enhancement methods and discuses their future development.

Abstrakt

Tato práce studuje 4 metody pro zvýrazňování detailů v digitálních fotografiích. Algoritmy metod jsou popsány a implementovány do stávajícího systému pomocí C++ a OpenCV. Metody jsou následně porovnány z hlediska časové a paměťové náročnosti a vyhodnoceny jsou také jejich výsledky pomocí uživatelského dotazníku. Práce obecně poskytuje přehled současných metod pro zvýraznění detailů ve fotografii a diskutuje jejich budoucí vývoj.

Keywords

image processing, photographic enhancement, detail enhancement, computational photography, C++, OpenCV

Klíčová slova

zpracování obrazu, vylepšování fotografií, zvýrazňování detailů, výpočetní fotografie, C++, OpenCV

Reference

HUDZIEC, Tomáš. *Photographic Detail Enhancement Methods.* Brno, 2019. Master's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Doc. Ing. Martin Čadík, Ph.D.

Photographic Detail Enhancement Methods

Declaration

Hereby I declare that this master's thesis was prepared as an original author's work under the supervision of Mr. Čadík. All the relevant information sources, which were used during preparation of this thesis, are properly cited and included in the list of references.

.....
Tomáš Hudziec
May 22, 2019

Acknowledgements

I want to thank my supervisor, Mr. Čadík, for sharing his expertise on topic during consultations and for his guidance. Also I thank company Red Hat Czech s.r.o. for renting computational hardware, which I used for work on this thesis.

Contents

Chapter 1

Introduction

Big progress in digital and photographic technologies in the last few decades brought new options, how photographs can captured and processed to get desired informations. Today digital image processing is used for various tasks and in diverse fields. To name few, there is object recognition in photograph or video, quality metrics of materials in industry process or architecture, determining diagnose from medical images, restoration of artworks and many more.

For good quality of photographs, contemporary cameras not only capture image, they also compute it. Thence here comes concept of *computational photography*. Its applications can be found on many of modern smartphones, e.g. features as HDR and panorama.

In fact, nowadays, almost everyone, who owns a smartphone, is a photographer. Huge amount of photographs is taken and being shared every day, but not all of taken pictures are of good quality, though. In big amount of pictures, there is often not much time to edit every single photograph manually, so there is a need for automatic or at least semi-automatic enhancing methods. Many of them are already implemented in photo-editing softwares. However, continuing research in this area brings improvements and new approaches every year.

This thesis studies few candidates of methods for detail enhancement in photographs, which were presented in research in recent years. First, field of photographic detail enhancement is introduced and its methods are categorized in chapter 2. Next chapter 3 contains description and theoretical principles for each method. Implementation of methods is described in chapter 4, where methods are implemented as plugins into existing software system. Chapter 5 contains comparison of methods' metrics, such as time and memory complexities and visual evaluation of their results via users questionnaire. In the last chapter 6 there is a conclusion of this work.

Chapter 2

Photographic Detail Enhancement

Reasons, why would we want to enhance images, are several. It can be desire to enhance have wrong quality of images due to wrong light conditions, fog or other reason. Images can be enhanced also for artistic purposes. One of studied methods is mentioned further is such artistic. It can be also more serious reason, like to get better diagnose in medical applications, such as radiograph or other image. In industry, enhancing photographs of manufactured products can help to detect defects on materials.

2.1 Digital image Representation

At first, let there be defined, what is an digital image. As computer works with numbers, digital image must be represented in numbers, so computer could work with it. Here helps mathematical definition.

2.1.1 Mathematical definition of grayscale image

Grayscale image can be defined as 2D function: $f : R^2 \to R$, where $f(x, y)$ gives image intensity at position (x, y) Realistically, we expect the image only to be defined over a rectangle, with a finite range:

$$f : [0 \,..\, m-1] \times [0 \,..\, n-1] \to [0, 1],$$

where m and n is width and height of image, respectively. Usually image intensity is normalized to interval $[0, 1]$, which represent energy of signal: 0 = zero energy is black and 1 = full energy is white. When speaking about energy, it is important to mention, that digital image can be also viewed as 2D signal.

Image 2D function rectangular space is sampled on a regular grid into units – pixels (picture elements) and then quantized – rounded to nearest discrete value representable by computer. Such rectangular matrix of discrete numbers is digital representation of image. Following figure 2.1 shows different views on digital image.

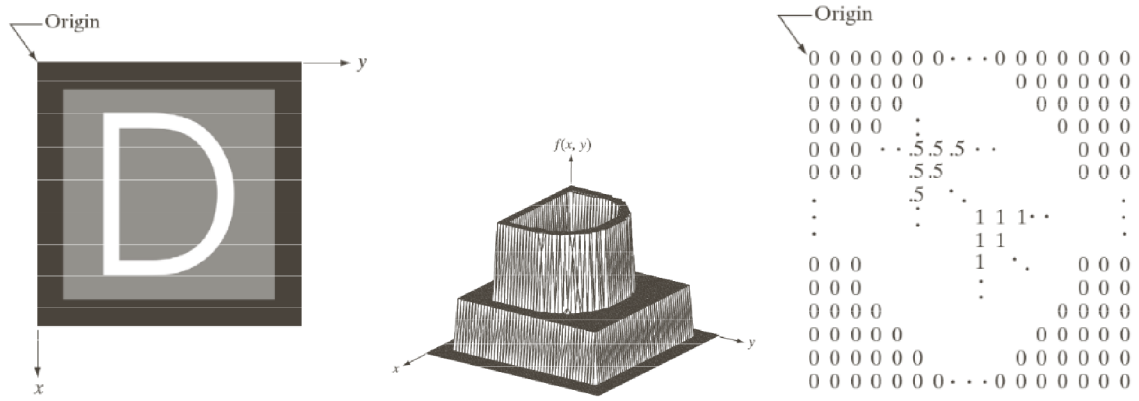


Figure 2.1: From left to right: Grayscale image ("viewed from upper"). Image plotted as a 2D function in 3D space. Image as 2D numerical array/matrix. Images taken from Digital Image Processing (DIP) book [13, p. 55].

2.1.2 Representing colours in images

Colours in image are represented by storing three values for red, green and blue components instead of one as in grayscale. With these three colours, any colour can be represented. Such image then has 3 channels, which can be stored as matrices. Other types than RGB also exists, though.

2.2 Digital image processing

Image processing, in most basic terms, is operation, that takes as input one image f, performs some computations on it and outputs another image g. Such processing can process intensities/ranges of each pixel separately – that is called point processing (range transformation):

$$g(x, y) = t(f(x, y))$$

Good example is creating image negative, where $t(x) = 1 - x$ for image range $[0, 1]$.

Some operations preserve pixel intensities, but change pixel positions:

$$g(x, y) = f(t_x(x, y), t_y(x, y))$$

where t_x and t_y are translation functions. Examples can be rotation, change of scale, warping – pixels are moved to another position.

2.3 Image details and their enhancing

For this thesis, which is concerned about photographic detail enhancement, it is important to define the details, which are going to be enhanced. By details are meant small changes in local contrast of image. For example some texture of wood or wall as is visible on figure 2.2.

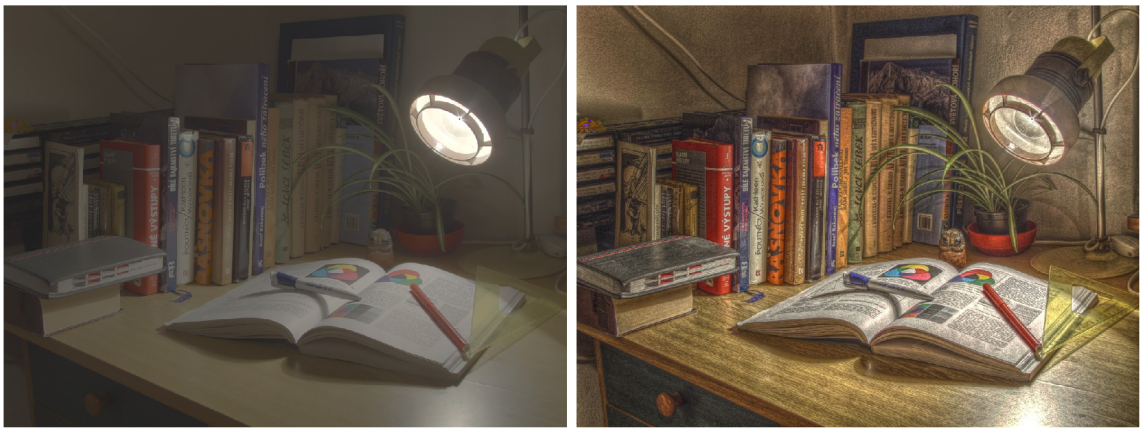


Figure 2.2: Example of detail enhanced image with method Local Laplacian Filtering [22]. Exaggeration of details on texture of wood and wall are particularly observable. Source of image is archive of Martin Čadík's testing HDR images available on his web page [18].

2.4 Image detail enhancement methods overview

There are more criteria for divisions of methods. This section gives overview of recent detail enhancement methods and provides their categorization based on different criteria.

2.4.1 Data vs. Algorithm orientation

First division can be into *data-driven* and *behaviour-driven* approaches.

Data-driven methods are more oriented on data (images in this case). They often employ some machine learning – training some internal model on big dataset of images and then they can apply learned knowledge on previously unseen image to get desired result. Authors Bychkovsky et al. used difference learning [6], and recently convolutional neural network was used by Gharbi et al. [12].

Behaviour-driven methods focus on procedures, how algorithm operates on given data and what computations are done. Following methods in this section are behaviour-driven.

2.4.2 Domain aspect

Another division is based on domain, in which method operates.

Spatial domain methods "The term *spatial domain* refers to the image plane itself, and image processing methods in this category are based on direct manipulation of pixels in an image." [13, p. 104]

Intensity transformations perform their operation on image pixels itself, pixel by pixel – e.g. contrast stretching method, which applies S-curve function to on every pixel [13, p. 106,115]. Example of S-curve is sigmoid function, which is used later in method #1, and there it will be defined. Here on figure 2.3 is shown, how sigmoid function stretches contrast of image.

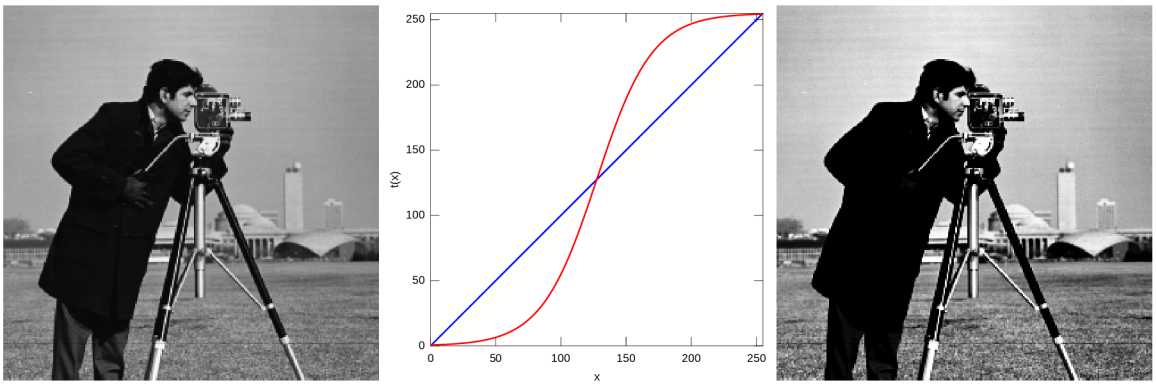


Figure 2.3: Contrast stretching with sigmoid function. From left: input image, sigmoid function, output image processed by sigmoid. Images taken from tutorial on intensity transforms of images [20].

Spatial filtering inferences output pixel from result of operation on neighborhood of input pixel – e.g. sharpen filter with Laplacian spatial mask [13, p. 160-162].

Histogram manipulation changes distibution of pixel intensities in image, it can be used for increasing contrast – e.g. various forms of histogram equalizations, such as [13, p. 122-144], [31].

Transform domain methods Processing does not have to be done on image itself, but also on its different representation, which is more suitable in some cases. Such methods transform an image into transform domain, there predefined operation is performed, and finally by applying inverse transform output image is obtained. Scheme of the process is shown at figure 2.4.

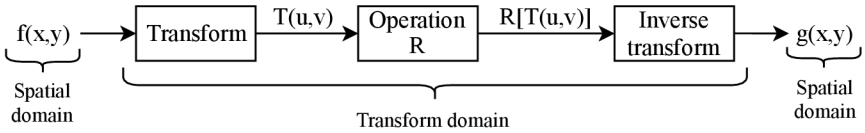


Figure 2.4: Diagram of image processing using transform domain [13, p. 94].

As a good example can serve filtering image in frequency domain, where discrete e.g. Fourier transform is used [13, p. 199]. Another example is logarithmic domain, which is better to work with high dynamic range images.

2.4.3 Edge-preserving decomposition methods

Being spatial domain based, these methods decompose input image into more layers and perform operations with them.

Most simple example works with 2 layers: Given *input image* I, *base layer* B is its smoothed version obtained by some smoothing algorithm which preserves significant edges, further described in section 2.4.4. *Detail layer* D is difference of input image and base layer: $D = I - B$. *Enhanced image* E is acquired, when scaled detail layer (multiplied by some constant) is added back to input image: $E = I + s \cdot D$.

Image edge-preserving decomposition methods can be divided into two categories: local filter based and global optimization based [14, p. 1].

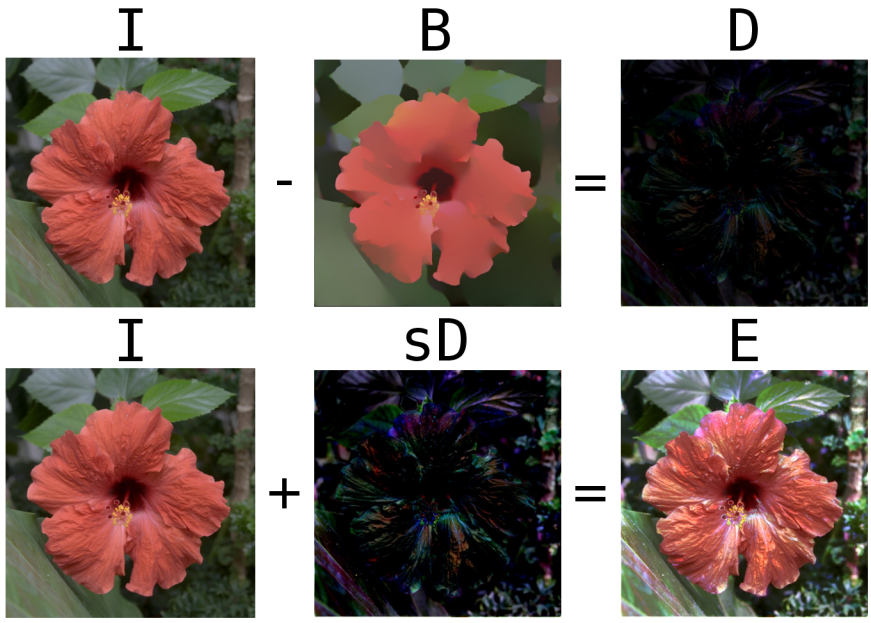


Figure 2.5: Basic approach for detail image enhancement. Scale coefficient s is set to 4 in this case. Detail layers are normalized for better visibility.

Local filter based Algorithms filtering image in local area are e.g. Bilateral filter [28] and local Laplacian filters [22].

Global optimization based These methods solve some global optimization problem, e.g. for smoothing [10], [29], or getting scale and shift coefficients for detail and base layer, respectively [26].

2.4.4 Edge-preserving smoothing

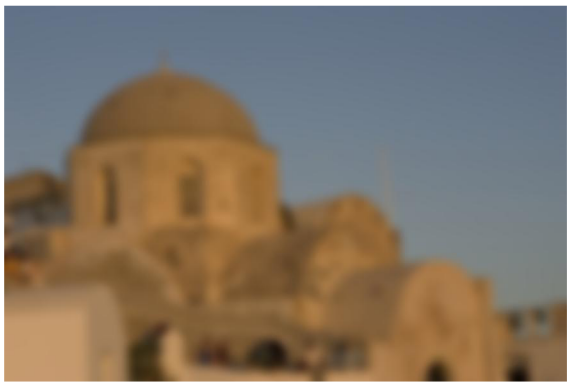
Smoothing in detail enhancement is used to produce base layer, as mentioned in previous section. Here may arise question on why such smoothing must be edge-preserving. For input image 2.6a results for different smoothing methods are shown.

Using Gaussian blurring on figure 2.6c, halo effect[1] in result around significant edges is introduced because Gaussian blurring does not preserve them. However, detail enhancement using edge-preserving smoothing on figure 2.6d produces halo-free result 2.6e, because the edges are preserved.

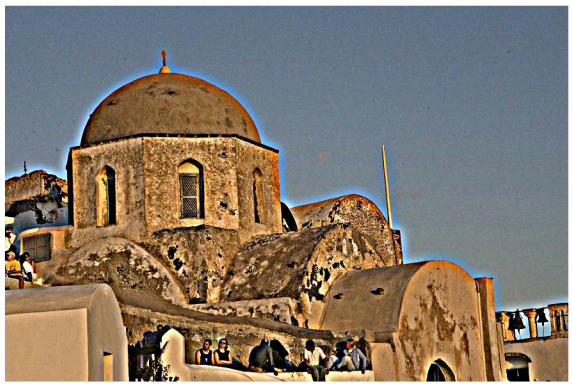
Therefore, smoothing used in detail enhancement pipeline must preserve significant edges in image.

[1] Artefacts around edges in image looking like halo (shine).

(a) Input image for smoothing [1]

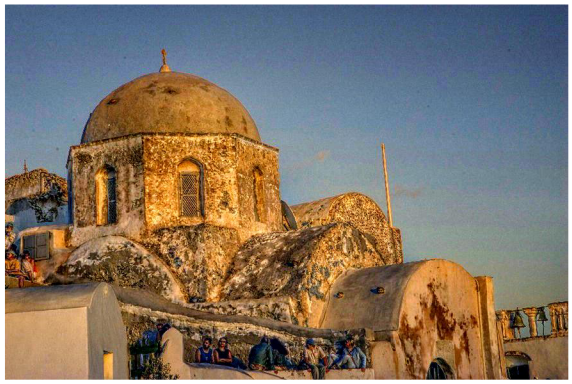


(b) Gaussian blurring with $\sigma = 8$



(c) Detail enhancement with halo effect

(d) L0 edge preserving smoothing [29]



(e) Halo-free result of detail enhancement

Figure 2.6: Gaussian blur is example of smoothing, which does not preserve edges (b). Therefore, in detail enhancement, blue halo effect is introduced around edges (c). When edge preserving smoothing is used (d), detail enhanced result is halo-free (e). Scale = 4 was used for detail enhancement. Example is exaggerated for obviousness. Original image is from results web page [1] for method discussed later [3].

2.4.5 Methods using image pyramids

Some methods [22], [3] use image pyramids – representation of image at different scales, originally described by authors Burt and Adelson [5]. As one method [3] will be studied in chapter 3, concept of these pyramids is presented below.

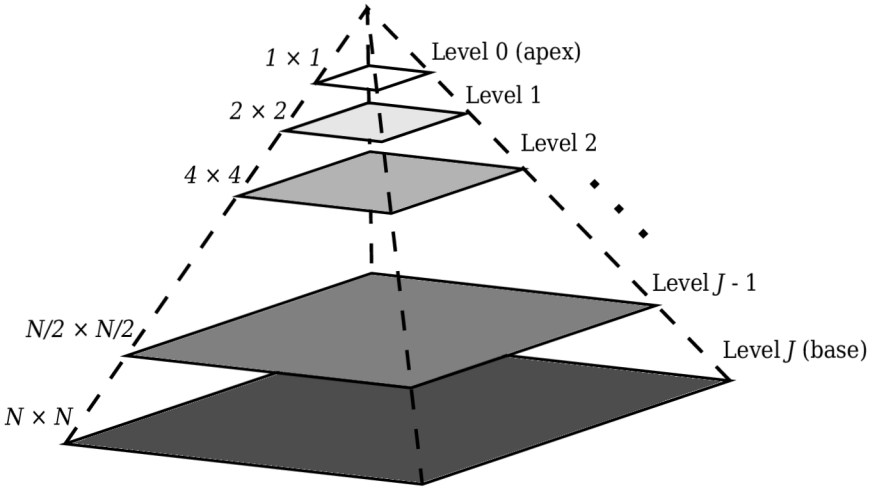


Figure 2.7: Illustrational scheme of image pyramids [13, p. 463]

Gaussian pyramid Starting with full resolution image, next level of Gaussian pyramid is made by blurring the image with Gaussian blur (approximation filter) and then scaling down by deleting every second row and column from image (downsampling) – left-top branch in pyramid system 2.10, result is approximation (on figure there should be level $j + 1$ above it).

Figure 2.8: Gaussian pyramid of example image. Taken from presentation on web of method #2 [2] and changed background.

Laplacian pyramid Laplacian is an operator of second derivative. In Laplacian pyramid, it is approximated with difference of Gaussians. Laplacian pyramid stores difference images between blurred versions of neighbor levels. Level j in pyramid is made by making difference between itself and downsampled and then upsampled self. Upsampling is done by inserting zero filled rows and columns after every second row/column. Interpolation filter can be also Gaussian blur – left-top-down-right path in pyramid system 2.10, result is prediction residual.

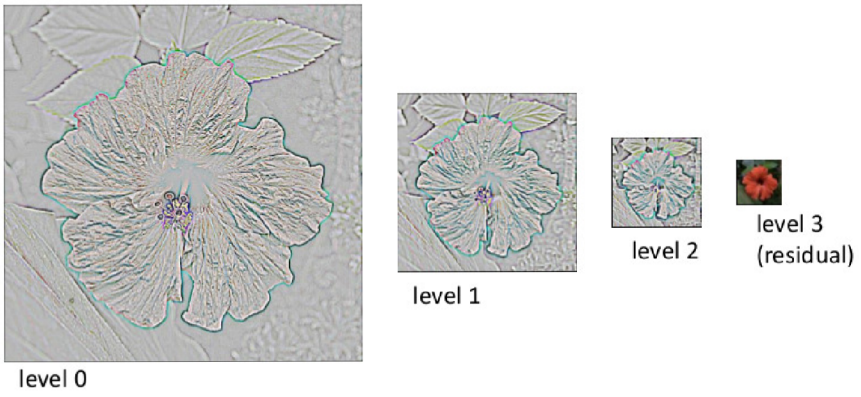


Figure 2.9: Laplacian pyramid of example image. Taken from presentation on web of method #2 [2] and changed background.

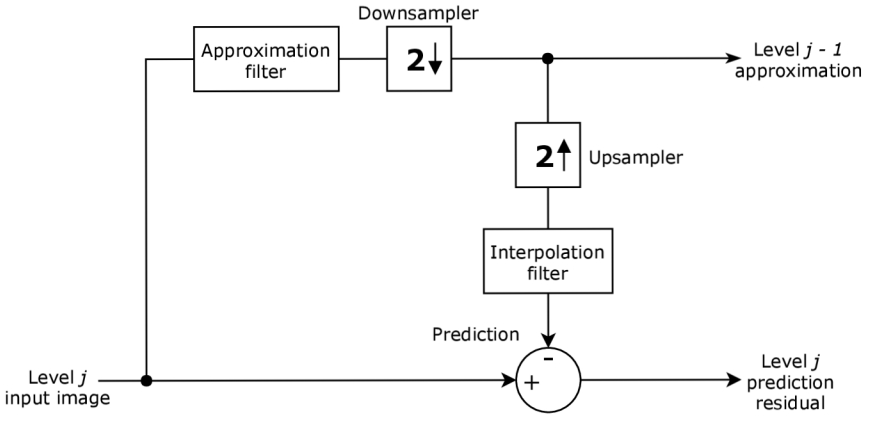


Figure 2.10: System for creating image pyramids [13, p. 463]

Chapter 3

Methods enhancing details in photographs

After introduction into topic and needed theoretical background, we can approach description of selected methods. At every method, principle and algorithm is described and also its result pictures are shown.

All following methods have similar structure – they take one image as input, perform some computation with it and output one enhanced image again. Before going to explain methods, here is common notation used in all methods:

- I – **I**nput **I**mage
- B – **B**ase layer (smoothed version of input image)
- D – **D**etail layer, $D = I - B$
- E – **E**nhanced output image

Subindex p means one pixel in image – e.g. I_p is pixel with index p in image I.

3.1 Edge-Preserving Decompositions for Multi-Scale Tone and Detail Manipulation

Described by authors Farbman et al. [10], this method introduces a way to construct edge-preserving multi-scale image decompositions, which is well suited for progressive coarsening[1] of images and for multi-scale detail extraction. Method introduces edge-preserving image smoothing operator using weighted least squares (WLS) optimization framework. This operator is used to construct edge-preserving multi-scale decompositions of image. These decompositions can be used for image detail enhancement.

3.1.1 Edge-preserving smoothing via WLS (Weighted Least Squares)

Base and detail layers are needed in this method to perform detail enhancing. Those are obtained by smoothing original image and other operations. Authors present edge-preserving smoothing approach based on the WLS optimization framework, which gives

[1] Image smoothing, which smoothes minor edges and preserves significant edges in image.

better results compared to linear Gaussian filter and Bilateral filter [10, p. 3]. Following paragraphs contain their description.

"Edge-preserving smoothing may be viewed as a compromise between two possibly contradictory goals. Given an input image I, we seek a new image B, which, on the one hand, is as close as possible to I, and, at the same time, is as smooth as possible everywhere, except across significant gradients in I." [10, p. 3]
Formally, this may be expressed as seeking the minimum of

$$\sum_p \left((B_p - I_p)^2 + \lambda \left(a_{x,p}(I) \left(\frac{\partial B}{\partial x} \right)_p^2 + a_{y,p}(I) \left(\frac{\partial B}{\partial y} \right)_p^2 \right) \right), \tag{3.1}$$

where the subscript p denotes the spatial location of a pixel. The goal of the *data term* $(B_p - I_p)^2$ is to minimize the distance between B and I, while the second (*regularization*) term strives to achieve smoothness by minimizing the partial derivatives of B. The smoothness requirement is enforced in a spatially varying manner via the *smoothness weights* a_x and a_y, which depend on I. Finally, λ is responsible for the balance between the two terms; increasing the value of λ results in progressively smoother images B.

3.1.2 Multi-scale edge-preserving decompositions

Using the edge-preserving operator described above, it is easy to construct a multi-scale edge-preserving decomposition. It consists of a coarse, piecewise smooth, version of the image, along with a sequence of difference images, capturing detail at progressively finer scales.

3.1.3 Multi-scale tone manipulation for detail enhancement

As said before, one of application of the method is image detail enhancement. It is done with multi-scale tone manipulation. Given an image, a three-level decomposition (coarse base level B and two detail levels D^1, D^2) of the CIELAB lightness channel is constructed.

Input parameters for the application are η – controlling exposure of base layer and boosting factors δ_0, δ_1, δ_2 for one base and two detail layers. The result of the manipulation E at each pixel p is then given by

$$E_p = \mu + S(\delta_0, \eta B_p - \mu) + S(\delta_1, D_p^1) + S(\delta_2, D_p^2), \tag{3.2}$$

where μ is the mean of the lightness range, and S is a sigmoid curve:

$$S(a, x) = \frac{1}{1 + e^{-ax}}, \tag{3.3}$$

which is appropriately shifted and normalized.

Similarly, layers of coarse, medium and fine detail boosting are computed. Final detail enhanced image is obtained as their mean average. All layers are shown in Figure 3.1.

Figure 3.1: Multi-scale tone manipulation using edge-preserving decompositions. Images taken from method's paper [10, p. 8].

3.2 Fast Local Laplacian Filters

Described by authors Aubry et al. [3], this method employs image pyramids, which concept was outlined in chapter 2.

3.2.1 Background on local Laplacian filters

This method is based on older "non-fast" method by authors Paris et al. [22].

Its algorithm is following: For each pixel in the Gaussian pyramid of the input (red dot), its value g_0 is looked up. Based on g_0, the input image is remapped using a pointwise function. Next, a Laplacian pyramid is build from this intermediate result, then the appropriate pixel is copied into the output Laplacian pyramid. This process is repeated for each pixel over all scales until the output pyramid is filled, which is then collapsed to give the final result. For more efficient computation, only parts of the intermediate pyramid need to be generated. Scheme of approach is depicted on figure 3.2.

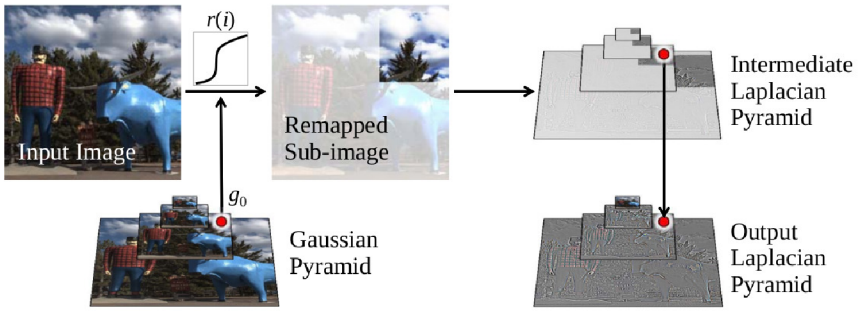


Figure 3.2: Local Laplacian Filters algorithm overview. Image taken from authors's paper [22, p. 5].

3.2.2 Design of the remapping function

Local Laplacian filters are used e.g. for tone mapping and detail manipulation. Authors proposed following remapping functions to compute the coefficient (ℓ, x, y) – which means pixel at position (x, y) in pyramid level ℓ:

$$\tilde{r}(I_p) = \begin{cases} g + sign(I_p - g)\sigma_r(|I_p - g|/\sigma_r)^\alpha & \text{if } I_p \leq \sigma_r \\ g + sign(I_p - g)(\beta(|I_p - g|/\sigma_r) + \sigma_r) & \text{if } I_p > \sigma_r \end{cases} \tag{3.4}$$

where: I_p is the pixel from input image, g is the coefficient of the Gaussian pyramid at (ℓ, x, y), which acts as a reference value, α controls the amount of detail increase $(0 \leq \alpha < 1)$ or decrease $(\alpha > 1)$, β controls the dynamic range compression $(0 \leq \beta < 1)$ or expansion $(\beta > 1)$, and σ_r defines the intensity threshold the separates details from edges. Sample functions are shown in figure 3.3.

3.2.3 Efficient local laplacian filtering

Authors propose an acceleration technique to evaluate local Laplacian filters on single-channel images, which encompasses also detail manipulation.

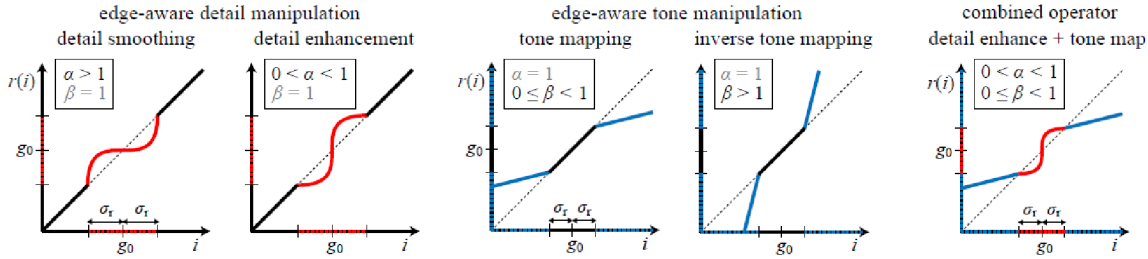


Figure 3.3: Family of point-wise remapping functions for edge-aware manipulation. Function plots obtained from authors's paper [22, p. 5].

Their strategy is based on the fact that the nonlinearity (remapping function) comes from the dependency on g (pixel from Gaussian pyramid). Characterization of this dependency in terms of signal processing allows them to design a theoretically grounded subsampling scheme that is more than an order of magnitude faster than the previous algorithm.

Fast method comes with following optimizations:

- number of precomputed pyramids is fixed, this algorithm has linear complexity in the number of pixels [3, p. 6]
- use of linear interpolation instead of a sinc kernel for reconstructing the signal [3, p. 6]

Optimized algorithm is 50x times faster than older method in SW and 10 times faster in HW [3, p. 6].

Figure 3.4: Left: original image. Right: detail enhanced image with Fast Local Laplacian Filters with parameters $\sigma = 0.5$, fact $= 6$, N $= 40$. Images taken from authors's results [1].

3.3 Art-Photographic Detail Enhancement

This "artistic" method, described by authors Son et al. [26], is inspired by principles of art photography, where exaggerated depiction of fine-scale detail is desirable and puts photography into aesthetic and artistic style.

The central idea, as described by authors, is "to introduce a piecewise smooth tone transform model to obtain extremely exaggerated local contrast for each region in the image (à la art photograph) while keeping the resulting tone within the target dynamic range." [26, p. 1]

3.3.1 Tone Transform Model

This section is based on authors's description from [26, p. 3]. Input image I is decomposed into base layer B and detail layer D, where B is obtained by smoothing I, and $D = I - B$. Let D_p denote the detail coefficient at each pixel p, that is, $D_p = I_p - B_p$. A simple way of enhancing detail would be to boost the detail coefficients D_p as follows:

$$E_p = B_p + D'_p = B_p + s_p D_p, \tag{3.5}$$

where s_p is a scale factor. However, the possible range of s_p is bounded by the input tone value at i because E_p cannot exceed the maximum dynamic range of the display device. This could greatly limit one's ability to enhance detail, especially in the already dark or bright regions. This problem is mitigated by the following modification to the tone transform model:

$$E_p = B_p + D'_p = (B_p + t_p) + s_p D_p, \tag{3.6}$$

where t_p denotes the amount of vertical shift applied to the base layer B_p. This model transforms the tone in two ways: t parameters are shifting the base layer and s parameters are scaling the detail layer. In a dark region of photograph, a positive t_p would brighten the base B_p and thus make possible for s_p to be larger. Similarly, a negative shift would be desirable in a highly bright region.

3.3.2 Image decomposition into base and detail layer

This section is based on authors's description from [26, p. 4].

Ideal base layer B obtained from I would have a constant tone within each homogeneous region while preserving the shape of the edges as closely as possible to those in I. In this method, L_0 smoothing [29] is performed on I to obtain B.

It is known that overly sharpened or blurred edges in B can cause visual artifacts near edges during detail enhancement. Since L_0 operation is designed to generate piecewise constant regions, it may produce many oversharpened edges.

This issue is addressed by classifying the sharpened edges into two groups: strong (hard) edges and smooth (soft) edges. The solution is composed of three passes and is designed to protect both types of edges from oversharpening:

- L_0 smoothing
- L_0 smoothing with adaptive λ map
- adaptive Gaussian blurring

L_0 smoothing

In the first pass, B^1 base layer is obtained by performing the original L_0 smoothing on I. The purpose of this pass is to build an adaptive λ-map to guide the second pass.

Originally presented by Xu et al. [29], L_0 smoothing minimizes L_0 norm gradient in image by solving following problem:

$$\min_B \left\{ \sum_i (B_p - I_p)^2 + \lambda \cdot C(B) \right\} \tag{3.7}$$

where $C(B) = \#\{i||\delta_x B_p| + |\delta_y B_p| \neq 0\}$ is a non-zero gradient counting function. This minimization is then solved via special alternating optimization with auxiliary variables.

Adaptive L_0 smoothing

The second pass performs L_0 smoothing on I again but this time using adaptive λ_p recorded in the λ-map. Since λ_p gets progressively smaller near the edges, this avoids oversharpening of strong edges and keeps their original shapes.

The adaptive λ-map used for the second pass is computed as follows. After the first pass, strong step edges can be easily detected from B^1 by thresholding G^1, the gradient of B^1. Edge pixels are determined via thresholding $G^1_p \geq a$ (default threshold $a = 0.2$ for all results), where λ_p is set as $\eta(\approx 0)$ to properly preserve the original edge shape without oversharpening. To avoid numerical errors a small η is used instead of zero. As there is a move away from edges, i.e., $G^1_p < a$, λ_p should rapidly increase up to λ to ensure piecewise flattening of edges within homogeneous region. This behavior is modeled with the minus half of integral bisquare function

$$\rho(u, \sigma) = \begin{cases} \frac{1}{3} & \text{if } u < -\sigma \\ \frac{u^2}{\sigma^2} - \frac{u^4}{\sigma^4} + \frac{u^6}{3\sigma^6} & \text{if } -\sigma \leq u < 0 \\ 0 & \text{if } u \geq 0 \end{cases} \tag{3.8}$$

The adaptive λ_p at pixel i is then defined:

$$\lambda_p = 3(\lambda - \eta)\rho(G^1_p - \alpha, \sigma) + \eta \tag{3.9}$$

with default value of $\sigma = 0.1$.
Let B^2 denote the outcome of this adaptive L_0 smoothing.

Adaptive Gaussian blurring

Finally, the edges in B^2 are adaptively Gaussian-blurred as described in [29]. The proper Gaussian scale at each pixel is measured by comparing the blurred version of B^2 and I. The outcome of this final smoothing serves as B.

Detail layer

Detail layer is then obtained by simply substracting base layer from original image, i.e. $D = I - B$.

3.3.3 Detail Maximization

Once the base and detail layers are ready, optimization of the tone transform parameters s_p and t_p can be done to bring out as much detail as possible. In this method, the process of detail enhancement is formulated as a constrained optimization problem, which is designed to maximize image detail while preserving scene structure.

Weight for piecewise control

The piecewise control of brightness in each region is enabled by introducing a weight $w_p \in [0, 1]$ that is inversely proportional to the gradient magnitude of the base layer B. The weight is defined as $w_p = K(\nabla B_p)$[2], where $K(u)$ denotes Tricube function:

$$w_p = K(\nabla B_p) = \begin{cases} (1 - |\frac{\nabla B_p}{a}|^3)^3 & \text{if } \nabla B_p \leq 1 \\ 0 & otherwise \end{cases} \tag{3.10}$$

Parameter a determines the value of u where w_p drops to zero. Default value of a is set to 0.2.

Detail measure

Given the decomposition of image $I = B + D$, the amount of detail in I is measured as the squared sum of all detail coefficients D_p in the detail layer D. Thus for the transformed image $E = sD + B + t$ in Eq. 3.6, the goal is to maximize $\sum_i \|s_p D_p\|^2$. To ensure piecewise smooth variation of s and t, the detail maximization scheme must simultaneously minimize the *smoothness terms* $\|\nabla s_p\|^2$ and $\|\nabla t_p\|^2$. This is to encourage region-based control of brightness and detail, inspired by the conventional process of art photography.

Objective function

Objective function to be minimized for maximizing detail is:

$$\begin{aligned} f(s,t) = &- \sum_i \|s_p D_p\|^2 \\ &+ r_1 \sum_i w_p \|\nabla s_p\|^2 + r_2 \sum_i w_p \|\nabla t_p\|^2, \end{aligned} \tag{3.11}$$

with constraint:

$$0 \leq E_p = (B_p + t_p) + s_p D_p \leq 1 \tag{3.12}$$

This constraint keeps output's image in tonal range [0,1] and s_p with t_p from going infinite. "Their smoothness terms further limit any drastic change of s_p and t_p in the local neighborhood except at region boundaries." [26, p. 5]

Authors chose to solve the minimization problem using quadratic problem solver – this is to obtain s_p and t_p that minimize $f(s,t)$. For convex problem condition, they constrain $r \cdot w_p$ to be larger than 2 by modifying w_p to $w_p = K(\nabla B_p) + 2/r$. To achieve piecewise constant scales and shifts, they use large values of r_1 and r_2 (with default value $r_1 = 200$ and $r_2 = 500$).

[2]Operator ∇ (nabla) means gradient operation, which can be computed as difference of neighbor elements.

Detail control via interpolation

Once the optimized image E has been obtained, it can be optionally used as an upper bound to control the amount of detail enhancement via linear interpolation with the input image I. Following equation shows the linear interpolation between images I and E using parameter $\mu \in [0, 1]$.

$$\begin{aligned} E_\mu &= \mu E + (1 - \mu) I \\ &= \mu(sD + B + t) + (1 - \mu)(D + B) \\ &= (\mu s + (1 - \mu))D + B + \mu t \end{aligned} \tag{3.13}$$

Figure 3.5 shows result of method Art-Photographic Detail Enhancement on input image.

Figure 3.5: Left: Original image. Right: Output image processed by the method. Notable is brightening of dark area, where shift coefficients t_p shifted base layer to brighter values. Images are from authors' results [25].

3.4 Content Adaptive Image Detail Enhancement

This method, described by authors Kou et al. [14], uses modified $L0$ norm gradient minimization algorithm, originally presented by Xu et al. [29]. The algorithm is used for image smoothing to get base layer of image. Here, the algorithm is modified to produce detail enhanced image directly.

3.4.1 Content adaptive detail enhance optimization

Minimizing $L0$ norm gradient in image is a global minimization problem.

"Enlarging the gradients of a source image is an effective method to sharpen the image. However, halo artifacts and gradient reversal artifacts could be produced if all the gradients of the source image are enlarged. To reduce such effects, only all the gradients except those of pixels at sharp edges are enlarged. Such an idea is formulated as an L_0 norm based global optimization problem to derive an appropriate. Same as existing global optimization problems, the proposed performance index consists of a data fidelity term and a regularization term. A lagrangian factor λ is used to adjust the importance of the two terms to control the degree of the enhancement. Based on these, the optimization problem is formulated as follows:" [14, p. 2]

$$\min_{E} \left\{ \sum_{p} (E_p - I_p)^2 + \lambda \cdot C(E - K \circ I) \right\}, \tag{3.14}$$

"where E is the detail-enhanced image, I is the input image, p is the pixel index of the images, $\circ$ denotes the element-wise product operator. For simplicity, we use $\hat{I}$ to stand for $K \circ I$. Note that $C(E - \hat{I})$ is the L_0 norm of the gradient field, which equals the number of non-zero elements of the gradient field of $E - \hat{I}$ defined" [14, p. 2]

$$C(E - \hat{I}) = \#\{p | \left|\partial_x (E_p - \hat{I}_p)\right| + \left|\partial_y (E_p - \hat{I}_p)\right| \neq 0\}, \tag{3.15}$$

"where # is a counting operator.

Without loss of generality, it is assumed that the detail layer is enhanced times in the final image. K_p is then computed as follows:" [14, p. 2]

$$K_p = 1 + \frac{k}{1 + e^{\eta \cdot (V_p - \overline{V}_p)}} \tag{3.16}$$

"where V_p is the variance of the pixels in the 3×3 neighborhood of the p-th pixel, $\overline{V}_p$ is the mean value of all the local variances. η is calculated as $ln(0.01)/(min(V_p) - \overline{V}_p)$, it guarantees the factors of small variance pixels be close to $1 + k$. So with 3.16, the factors of large variance pixels are close to 1." [14, p. 2]

Results of this method can be seen on figure 3.6.

Figure 3.6: Original image (left) and enhanced image with Content Adaptive Image Detail Enhancement method (right). Original image as in author's paper [14, p. 3] and enhanced image computed from the original with the code.

3.5 Deep bilateral learning for real-time image enhancement

The last studied method in this thesis is described by authors Gharbi et al. [12]. It is based on training neural network on pairs of input and output images, between which there can be arbitrary transform operation, even human retouch. Treating transform operation as black box, neural network will learn it from differences of gradients between input and output images through backpropagation. Then, given some test unseen image, it can reproduce learned edit operation.

This method was added rather as representative of novel data-driven approach, so only outline of architecture is here provided, as described by authors. However, this method has potential for future development of image editations.

3.5.1 Architecture outline

"Network architecture of this method seeks to perform as much computation as possible at a low resolution, while still capturing high-frequency effects at full image resolution. It consists of two distinct streams operating at different resolutions. The *low-resolution* stream (top) processes a downsampled version $\tilde{I}$ of the input I through several convolutional layers so as to estimate a bilateral grid of affine coefficients A. This low-resolution stream is further split in two paths to learn both local features L^i and global features G^i, which are fused (F) before making the final prediction. The global and local paths share a common set of low-level features S^i. In turn, the *high-resolution* stream (bottom) performs a minimal yet critical amount of work: it learns a grayscale guidance map g used by our new *slicing* node to upsample the grid of affine coefficients back to full-resolution $\bar{A}$. These per-pixel local affine transformations are then applied to the full-resolution input, which yields the final output O." [12, p. 4]

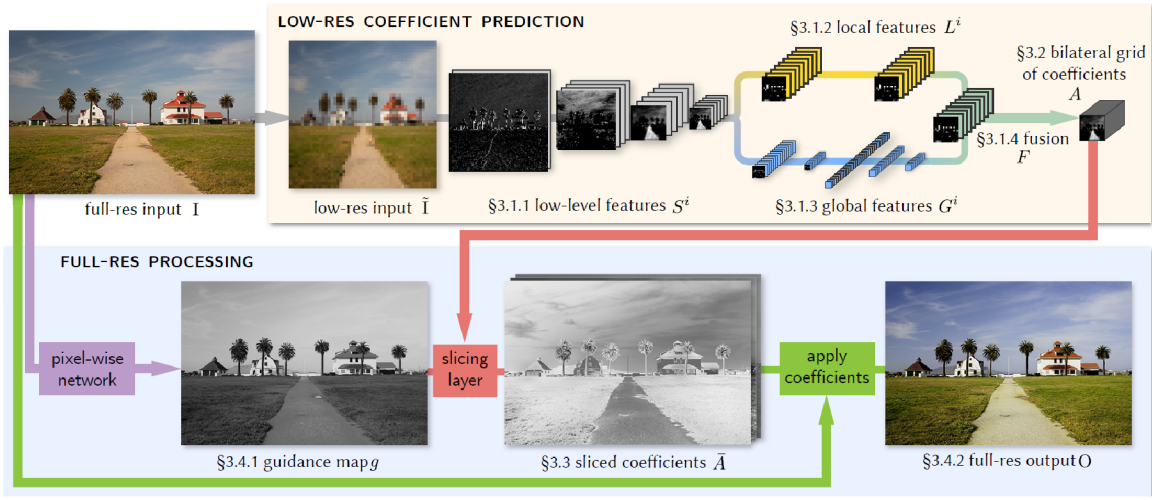


Figure 3.7: Method's system architecture. Image taken from method's paper [12, p. 4].

Chapter 4

Implementation of photographic detail enhancement methods

4.1 Tone Mapping Studio

Methods are implemented as plugins into existing software system – Tone Mapping Studio (TMS) [19]. It is a plugin-based framework written in C++ for image processing operations, mainly for HDR tone mapping and color-to-grayscale conversions. Its main author, Martin Čadík, has also done studies on evaluation of these types of methods [8], [7]. Other types of methods are also implemented in TMS, such as image detail enhancement methods. The program with its graphical user interface – `tmogui` – is shown on figure 4.1.

TMS also provides command line program – `tmocmd`, which is particularly suitable for usage in scripts.

4.2 Edge-Preserving Decompositions for Multi-Scale Tone and Detail Manipulation

To use this method for detail enhancement, I followed recipe for multi-scale tone manipulation application described in the paper [10, p. 7]. On webpage for this method [17], Matlab code is available, which was used as reference for implementation.

Original implementation consists of WLS optimization for image smoothing. In my implementation I used instead of WLS smoothing Fast Global Smoothing method. It is 30x faster than WLS [21, p. 10] and also it is integrated in OpenCV library as `cv::ximgproc::FastGlobalSmootherFilter` Class. Parameters were set empirically, so that smoothing results would match with those of authors.

From input image a three-level decomposition are constructed: coarse base layer B and two detail layers D^1 and D^2. Each of this layers is passed to sigmoid function with different parameters – boosting factors. Output image is obtained by summing result from sigmoid function. Also parameters `gamma`, `exposure`, `saturation` are involved.

Method algorithm implementation steps

- smooth input RGB image twice by FGS filter to get two smoothed versions: less and more smoothed

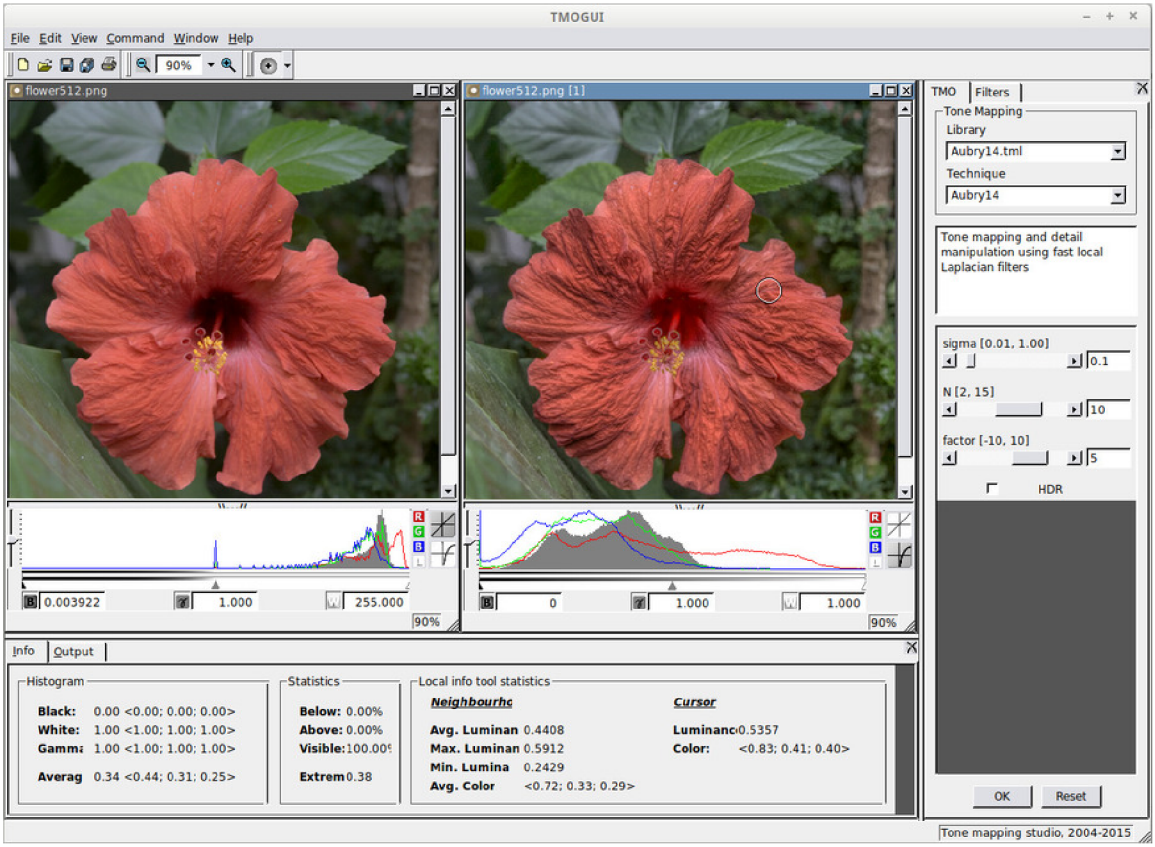


Figure 4.1: Screenshot of Tone Mapping Studio.

- convert 2 smoothed versions to Lab color space and get their luminance channels: $L0$ and $L1$
- call `tonemapLAB` function 3 times with different parameters to get sub-results with fine, medium and coarse details enhanced
- make average from these sub-results to get final result

tonemapLAB function steps

- calculate first difference image D^1 by calling sigmoid function on $L - L0$
- calculate second difference image D^2 by calling sigmoid function on $L0 - L1$
- calculate base image by calling sigmoid function on L1 adjusted by exposure and value 56 (= mean of lightness range)
- luminance result is sum of difference images and base image

In my implementation user can choose, from which sub-result images final result will be composed: it can be one or more from coarse, medium and fine detail enhanced. For each layer user can choose parameters gamma, exposure, saturation and also boosting factors $\delta_0 - \delta_2$.

On figure 4.2 is comparison of my result with authors'. Authors' result has more contrast in flower area, but also more blurred background. Otherwise are results similar. More identical result to authors' could be produced, if the same smoothing would be used.

Figure 4.2: From left to right: original image, authors' result, my result.

4.3 Fast Local Laplacian Filters

Implementation for this method is done for LDR images as my project from previous course on Computational Photography. My goal in this thesis is to extend it to work with HDR images. Method's algorithm is as follows:

- Make grayscale version of input image
- Calculate colours ratio as i for converting to rgb at the end.
- Extend image with border to make it of size of 2^n for pyramid functions.
- Main algorithm on grayscale version of image with use of OpenCV with pyramids: Gaussian Pyramids, Laplacian Pyramids
- get colours back by multiplying result with color ratios

For image pyramid functions I use OpenCV functions `cv::pyrUp()` and `cv::pyrDown()`.

My contribution on method in this thesis:

- Conversion of input HDR image into logarithmic scale and after main algorithm computation converting it back to normal scale.
- Implementation of HDR postprocessing.
- Moving main algorithm code into a function.

For implementing HDR support I also extended authors' fast method's code for HDR (it was only at original method). On figure 4.3 are shown results. My result is darker and have less details enhanced than fast authors' version, but certainly is better than simple gamma tone mapped image. The biggest amount of detail enhances authors' original method, but it is a lot more time expensive (about 50x).

(a) Input gamma tone mapped image

(b) My result of tone mapping

(c) Authors' fast method [3]

(d) Authors' original method [22]

Figure 4.3: Comparison of results. Source of input HDR image is web page for paper on HDR Companding [16]. Others were generated using codes.

4.4 Art-Photographic Detail Enhancement

Most of implementation of this method was done by Pavel Sedlář in his bachelor thesis [24]. However, it could only handle very small images (to 32x32 pixels) due to huge memory consumption of program. Also, results does not seem the same as at original implementation of authors. My goal was to extend abilities of current implementation to handle bigger images by using sparse matrices and also try to correct enhancing functionality. Also computational time even for small image as 32x32 pixels is about 14 minutes on HP Probook 4540s 1.9GHz Intel Celeron B840 with 4GB RAM.

4.4.1 Algorithm overview

To briefly remind structure of algorithm, here is an overview of it:

- Base Decomposition
  - L0 smoothing of input image I
  - adaptive L0 smoothing
  - adaptive Gaussian blur smoothing (not implemented)
  - output is base layer B
- Detail Maximization
  - obtaining detail layer as $D = I - B$
  - getting weights for piecewise control of brightness and detail
  - optimization of objective function to get scale and shift coefficients s and t
  - getting output enhanced image as $E = (B + t) + sD$

4.4.2 Base decomposition

Smoothings for base decomposition are described in section 3.3.2. Implementation of first two steps (L_0 smoothings) was done by previous author as described in paper.

4.4.3 Detail maximization

As already said, detail maximization is done by optimization of objective function to get scale and shift coefficients. This maximization is computed with quadratic programming (QP). It is a mathematical optimization problem with linear constraints, which solves following equation:

$$\begin{aligned} \min_{x} \quad & x^T H x + c^T x + c_0 \\ \text{s.t.} \quad & Ax \leq b \\ & l \leq x \leq u \end{aligned}$$

for n-dimensional variables vector x and m constraints, where

- H is $n \times n$-dimensional real symmetric matrix also called Hessian matrix, which defines terms in quadratic equation
- c is n-dimensional real-valued vector for linear programming

- c_0 is a constant
- A is an $m \times n$ real matrix of constraints
- b is m-dimensional real vector with constraints' bounds
- l is n-dimensional vector of lower bounds for variables
- u is n-dimensional vector of upper bounds for variables

For C++ programming language, there are lots of QP solvers, which can be used to solve QP problems (also called as QP). Current solution used library qpOASES [11] – QP solver based on a parametric active-set method. This method can effectively use *a priori* information to speed-up computation of a QP solution, but its computational complexity grows exponentially with the number of constraints [4, p. 1-2]. It has been originally developed for small to medium scale QPs featuring dense Hessian and constraint matrices.

However, QP in this method is large scale, because for each pixel in image we need two variables – shift and scale parameters. Thus, for square image with edge size N, Hessian matrix is of size $2N^2 \times 2N^2$. Even for very small image of size 256×256 Hessian matrix will contain $(2 * 256^2)^2 = 17.18 \times 10^9$ elements. Using 4-byte `float` type for storing real numbers, the Hessian would settle 64GB in memory, which is huge amount for today personal computers for such small picture.

Good news are, that Hessian matrix is very sparse, because smoothness terms in objective function 3.11 contain only neighbor elements (from gradient operator ∇), which in Hessian will result in non-zero elements near to diagonal, and zero elements elsewhere. Therefore, it is very desirable to use sparse matrix type to store Hessian matrix.

qpOASES additionally comes with limited support of sparse matrices, so my first steps led this way. Sparse matrix type here is stored in compressed sparse column (CSC) format. It represents a matrix by three (one-dimensional) arrays, that respectively contain nonzero values, row indices, and the column pointers to first elements in columns.

So I implemented algorithm for converting triplets of non-zero entries of sparse matrix (x, y, value) to arrays of CSC format. This algorithm first sorts non-zero elements to column-major order and then fills appropriately CSC arrays. From those arrays qpOASES can build sparse matrix.

Memory for storing sparse matrix was reduced, it was good step. However, when I tried to launch QP solver on 64×64 image, memory usage grew to 3GB.

In spite of sparse matrix support, linear algebra in qpOASES is implemented in dense fashion. So from there comes this high memory consumption. Also computational time have not decreased.

One more chance for qpOASES was external solver MA57 [9] written in Fortran. After obtaining personal license and compilation, sadly problems with undefined symbol error were met at linking time. I made decision to look after some other QP solving library.

As candidate was chosen OSQP [27], open-source operator splitting QP solver written in the C language, with interfaces to high-level languages including Matlab, Python and Julia. Most importantly for this method, it contains custom sparse linear algebra routines, which can exploit sparsity of input matrices.

OSQP use similar interface to qpOASES, so transition was not complicated. Results are very good, since for 32x32 image computation time is 7s, which is 120x faster, than with previous solver. Largest tried image was 350x272, on which computation took 5m 22s and 570MB on Lenovo X1 Carbon 4th Gen 2.50GHz Intel Core i7 with 8GB RAM.

On figure 4.4 there are results, both authors' and mine. My result has changed colours and brightness, and it is not as expected. But compared to previous state, significant speedup was reached and bigger images were allowed. Structure of image is preserved, but it would require yet some fine tuning and debuging, until result will be similar to authors'.

Figure 4.4: From left to right: original image, author's result, my result. Source of original and authors' images is authors' result web page [25].

4.5 Content Adaptive Image Detail Enhancement

This method uses modified L_0 norm gradient minimization algorithm, which was originally used for image smoothing [29]. I used C++ implementation of original L_0 norm gradient minimization algorithm available from github [30] and modified it inspired by author's modified version in MATLAB from author's web page [15], so implementation was straightforward. Modification consists of additional calculating of layer with variances of pixels in the 3×3 neighbourhood. From that variance is then computed weight with sigmoid function, which is then used to modify computing of arbitrary variables h, v while solving global optimization problem – L_0 norm gradient minimization. Output enhanced image is direct result of this modified algorithm.

On figure 4.5 is compared author's and my result. My result has not as vivid colours as author's, but enhances details in similar way.

Figure 4.5: Detail enhancement using Content Adaptive Image Detail Enhancement method. From left to right: original image, author's result, my result. Image was taken from archive with reference source code from author's web page [15].

Chapter 5

Comparison and evaluation

For comparison, methods will be marked by their main author and year:

- Farbman08 – Edge-Preserving Decompositions for Multi-Scale Tone and Detail Manipulation
- Aubry14 – Fast Local Laplacian Filters
- Son14 – Art-Photographic Detail Enhancement
- Kou15 – Content Adaptive Image Detail Enhancement

5.1 Methods' metrics

First comparison of methods is their performance, which is time and memory complexities.

5.1.1 Time and memory complexities

In following list methods are sorted from fastest method to slowest:

- Farbman08 – 0.7 second on 350x250 image
- Aubry14 – 2 seconds on 350x250 image
- Kou15 – 10 seconds on 350x250 image
- Son14 – 5 minutes on 350x250 image

In following list methods are sorted from least memory demand to biggest:

- Farbman08 – 9.5MB on 350x250 image
- Aubry14 – 24MB on 350x250 image
- Kou15 – 31MB on 350x250 image
- Son14 – 500MB on 350x250 image

5.2 User's visual questionnaire

The second comparison aims on methods' results. Here, in users' questionnaire, participants rated methods' results, how they like them, on 12 chosen input images, which are shown in table 5.1. Original Matlab codes were used to generate images for questionnaire, except for 3.method, where we used our own Matlab code (which was kindly provided to me by supervisor and edited by me afterwards). For 3.method was used RGB version with central differencies and results was histogram-equalized, except portraits. Questionnaire was created with online service [23].

The questionnaire completed 308 respondents, from which were **70.78**% men and **29.22**% women.

On figure 5.1 there are global scores for each method through all 12 images.

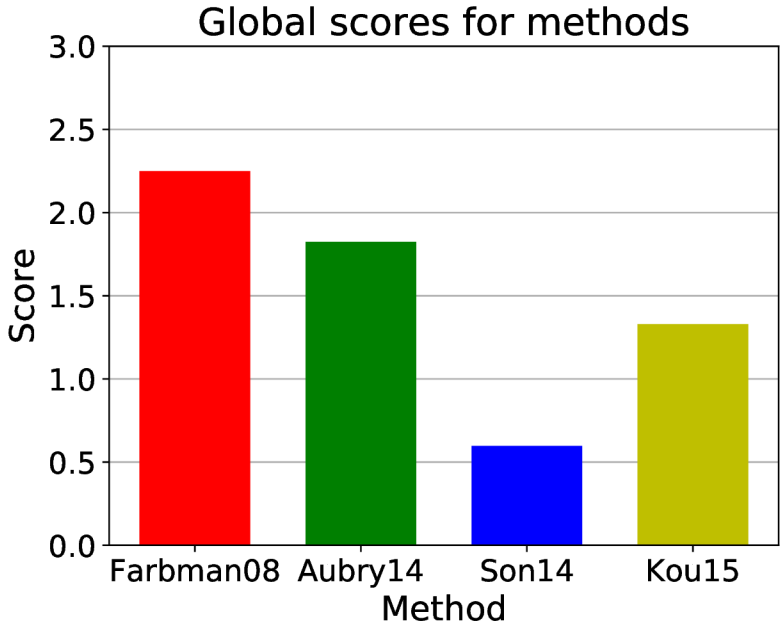


Figure 5.1: Chart of means of all scores.

In appendix A there are scores of methods for each of 12 images. Enhanced images itself are also shown.

5.2.1 Evaluation of questionnaire results

Method Farbman08 won. I guess it is because it made very little changes to original images compared to others. The worst rating had method Son14. I think it is because method was not exactly the same way implemented compared to original code and result was not correct.

One important information from questionnaire is, that detail enhancement methods are not suitable for editing portraits of people, and probably also animals. It is because people want to have smooth skin on photographs, and not enhanced every pigment and imperfection.

01 	02 	03 
04 	05 	06 
07 	08 	09 
10 	11 	12 

Table 5.1: Images used for questionnaire. Images 06 and 07 are „Designed by rawpixel.com / Freepik“.

Chapter 6

Conclusion

The goal of this thesis was to study and compare five recent photographic detail enhancement methods. Let's summarize characteristics of them.

The first method was based on edge-preserving decompositions of image. One of its applications was tone manipulation, which was also implemented. The most complex part was smoothing based on Weighted Least Squares optimization. It was replaced by more recent and faster Fast Global Smoothing algorithm, but WLS implementation is planned for future work.

The second method worked with Laplacian pyramids, was based on previous similar study, but brought optimization and speedup. It was based on subsampling, which reduced computational cost and approximated well exact solution. Except detail enhancement, the method also did good tone mapping.

The third method was artistic, and could bring much details even from dark or bright regions of picture. It was because of shift coefficients of base layer, which method introduced instead of only scales of detail layer. Computing these coefficients was, however, very expensive, because of need for solving global optimization problem with huge amount of variables.

The fourth method used for detail enhancement adjusted L_0 norm gradient minimization, which was originally used for image smoothing. It is also global optimization problem, but performance is yet unknown, because method was not yet implemented.

The last method is the most recent one among others. It uses very different approach. It is data-driven and employs machine learning with convolutional neural network. Although training requires some time, very big advantage is, that it can adapt on almost any image editation, even human retouch. It is also very fast – it runs on mobile hardware in real time.

For future development I see trend, which is going more to data-driven approaches, than exact mathematical algorithms. It can be predicted, that machine learning will be used more and more in digital image enhancement.

Bibliography

[1] Aubry, M.; Paris, S.; Hasinoff, S. W.; et al.: Fast Local Laplacian Filters: Theory and Applications, Large detail enhancement – additional results.
Retrieved from: http://imagine.enpc.fr/~aubrym/projects/llf/supplementary_material/additional_results.html

[2] Aubry, M.; Paris, S.; Hasinoff, S. W.; et al.: Fast Local Laplacian Filters: Theory and Applications, Large detail enhancement – web page.
Retrieved from: http://imagine.enpc.fr/~aubrym/projects/llf/

[3] Aubry, M.; Paris, S.; Hasinoff, S. W.; et al.: Fast Local Laplacian Filters: Theory and Applications. *ACM Transactions on Graphics*. vol. 33, no. 5. 2014: pp. 167:1–167:14. ISSN 0730-0301. doi:10.1145/2629645.
Retrieved from: http://doi.acm.org/10.1145/2629645

[4] Banjac, G.; Stellato, B.; Moehle, N.; et al.: Embedded code generation using the OSQP solver. In *IEEE Conference on Decision and Control (CDC)*. December 2017. doi:10.1109/CDC.2017.8263928.
Retrieved from: https://doi.org/10.1109/CDC.2017.8263928

[5] Burt, P.; Adelson, E.: The Laplacian Pyramid as a Compact Image Code. *IEEE Transactions on Communications*. vol. 31, no. 4. April 1983: pp. 532–540. ISSN 0090-6778. doi:10.1109/TCOM.1983.1095851.

[6] Bychkovsky, V.; Paris, S.; Chan, E.; et al.: Learning Photographic Global Tonal Adjustment with a Database of Input / Output Image Pairs. In *The Twenty-Fourth IEEE Conference on Computer Vision and Pattern Recognition*. Colorado Springs, CO. June 2011.

[7] Čadík, M.: Perceptual Evaluation of Color-to Grayscale Image Conversions. *Comput. Graph. Forum*. vol. 27, no. 7. 2008: pp. 1745–1754.

[8] Čadík, M.; Wimmer, M.; Neumann, L.; et al.: Evaluation of HDR Tone Mapping Methods Using Essential Perceptual Attributes. *Computers & Graphics*. vol. 32. 2008: pp. 330–349. ISSN 0097-8493.
Retrieved from: http://cadik.posvete.cz/tmo/cadik08cag.pdf

[9] Duff, I. S.: MA57—a Code for the Solution of Sparse Symmetric Definite and Indefinite Systems. *ACM Trans. Math. Softw.*. vol. 30, no. 2. June 2004: pp. 118–144. ISSN 0098-3500. doi:10.1145/992200.992202.
Retrieved from: http://doi.acm.org/10.1145/992200.992202

[10] Farbman, Z.; Fattal, R.; Lischinski, D.; et al.: Edge-preserving Decompositions for Multi-scale Tone and Detail Manipulation. In *ACM SIGGRAPH 2008 Papers*. SIGGRAPH '08. New York, NY, USA: ACM. Aug 2008. ISBN 978-1-4503-0112-1. pp. 67:1–67:10. doi:10.1145/1399504.1360666.
Retrieved from: http://doi.acm.org/10.1145/1399504.1360666

[11] Ferreau, H.; Kirches, C.; Potschka, A.; et al.: qpOASES: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*. vol. 6, no. 4. 2014: pp. 327–363.

[12] Gharbi, M.; Chen, J.; Barron, J. T.; et al.: Deep bilateral learning for real-time image enhancement. *ACM Transactions on Graphics (TOG)*. vol. 36, no. 4. 2017: page 118.

[13] Gonzalez, R. C.; Woods, R. E.: *Digital Image Processing (3rd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc.. 2006. ISBN 013168728X.

[14] Kou, F.; Chen, W.; Li, Z.; et al.: Content Adaptive Image Detail Enhancement. *IEEE Signal Processing Letters*. vol. 22. 2015: pp. 211–215.

[15] Kou Fei: KOU Fei's homepage.
Retrieved from: http://koufei.weebly.com

[16] Li, Y.; Sharan, L.; Adelson, E. H.: Compressing and Companding High Dynamic Range Images with Subband Architectures.
Retrieved from: http://www.mit.edu/~yzli/hdr_companding.htm

[17] Lischinski, D.: Edge-Preserving Decompositions for Multi-Scale Tone and Detail Manipulation [webpage].
Retrieved from: http://www.cs.huji.ac.il/~danix/epd/

[18] Martin Čadík: Evaluation of tone mapping operators.
Retrieved from: http://cadik.posvete.cz/tmo/

[19] Martin Čadík, Ondrej Hajdok, Michal Augustyn, Ondrej Fialka, Antonin Lejsek, Petr Bilek, Ondrej Pecina, Pavel Fryz, Martin Molek, Vladimir Vlkovic, Jan Brida, Petr Pospisil, Tomas Chlubna, Filip Brezna, Tomas Hudziec: Tone Mapping Studio (TMS).
Retrieved from: https://github.com/cadik/TMS

[20] matlabtricks.com: A tutorial on intensity transforms of images.
Retrieved from: https://matlabtricks.com/post-45/a-tutorial-on-intensity-transforms-of-images

[21] Min, D.; Choi, S.; Lu, J.; et al.: Fast Global Image Smoothing Based on Weighted Least Squares. *IEEE Transactions on Image Processing*. vol. 23, no. 12. Dec 2014: pp. 5638–5653. ISSN 1057-7149. doi:10.1109/TIP.2014.2366600.

[22] Paris, S.; W. Hasinoff, S.; Kautz, J.: Local Laplacian Filters: Edge-aware Image Processing with a Laplacian Pyramid. *ACM Transactions on Graphics*. vol. 30. Jul 2011: page 68. doi:10.1145/2010324.1964963.

[23] QuestionPro Survey Software: SURVEY ANALYTICS.
Retrieved from: https://www.surveyanalytics.com

[24] Sedlář, P.: *Metody zvýrazňující detaily ve fotografii*. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. 2018.
Retrieved from: http://www.fit.vutbr.cz/study/DP/BP.php?id=21158

[25] Son, M.; Lee, Y.; Kang, H.; et al.: Art-Photographic Detail Enhancement – More Results.
Retrieved from: http://cg.postech.ac.kr/research/art_photograph/results.php

[26] Son, M.; Lee, Y.; Kang, H.; et al.: Art-Photographic Detail Enhancement. *Computer Graphics Forum*. vol. 33, no. 2. 2014: pp. 391–400. doi:10.1111/cgf.12298.

[27] Stellato, B.; Banjac, G.; Goulart, P.; et al.: OSQP: An Operator Splitting Solver for Quadratic Programs. *ArXiv e-prints*. November 2017. 1711.08013.

[28] Tomasi, C.; Manduchi, R.: Bilateral filtering for gray and color images. In *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*. Jan 1998. pp. 839–846. doi:10.1109/ICCV.1998.710815.

[29] Xu, L.; Lu, C.; Xu, Y.; et al.: Image Smoothing via L0 Gradient Minimization. *ACM Transactions on Graphics (SIGGRAPH Asia)*. 2011.

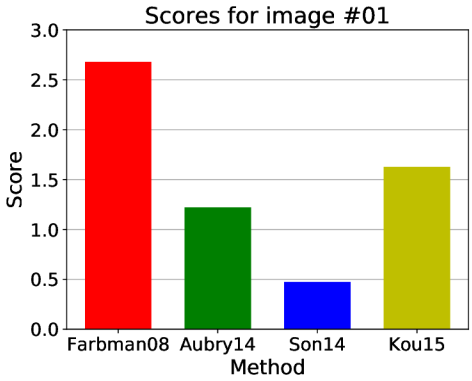
[30] Yamanaka, D.: A C++ implementation of Image Smoothing via L0 Gradient Minimization.
Retrieved from: https://github.com/daikiyamanaka/L0-gradient-smoothing

[31] Zuiderveld, K.: Graphics Gems IV. chapter Contrast Limited Adaptive Histogram Equalization. San Diego, CA, USA: Academic Press Professional, Inc.. 1994. ISBN 0-12-336155-9. pp. 474–485.
Retrieved from: http://dl.acm.org/citation.cfm?id=180895.180940

Appendix A

Results of questionnaire

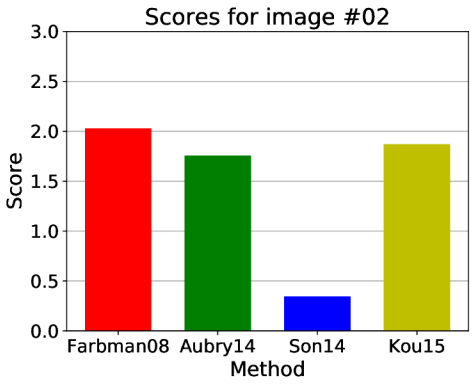
Table A.1: This table presents results of questionnaire, where respondents rated methods' results on 12 photographs. Bar charts show score for each method on particular image, the bigger score means better rating. Results for method Son14 were histogram-equalized, except for human portraits.

Questionnaire results on methods' comparison	
Original image #01	Scores of methods' results
	
 Farbman08	 Aubry14
 Son14	 Kou15

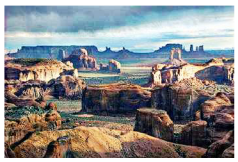
Continuation of questionnaire results on methods' comparison A.1

Original image #02

Scores of methods' results



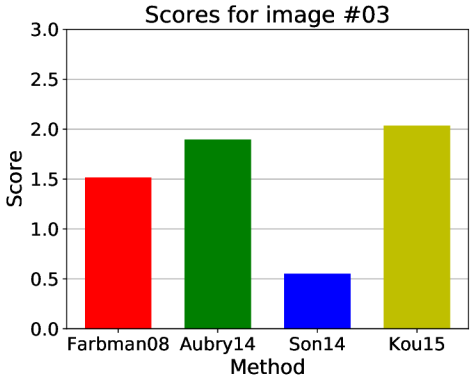





Farbman08 Aubry14 Son14 Kou15

Original image #03

Scores of methods' results



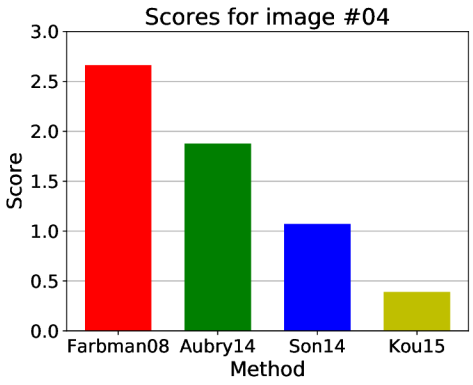



Farbman08 Aubry14 Son14 Kou15

Continuation of questionnaire results on methods' comparison A.1

Original image #04

Scores of methods' results




Farbman08

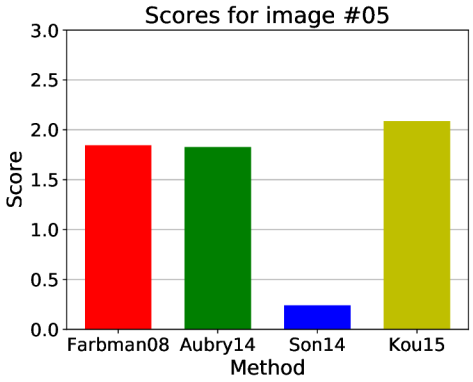

Aubry14


Son14


Kou15

Original image #05

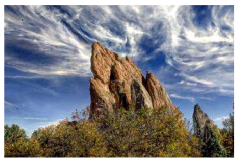
Scores of methods' results




Farbman08


Aubry14

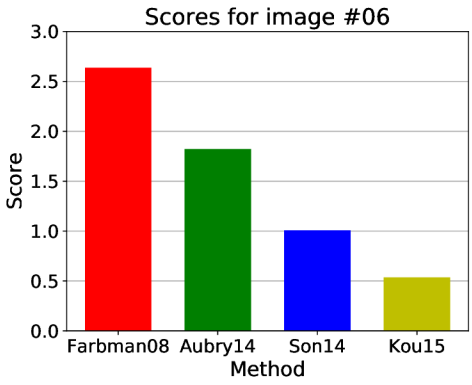

Son14


Kou15

Continuation of questionnaire results on methods' comparison A.1

Original image #06

Scores of methods' results




Farbman08

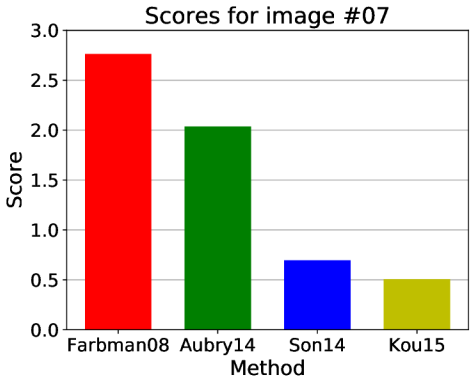

Aubry14


Son14


Kou15

Original image #07

Scores of methods' results




Farbman08


Aubry14

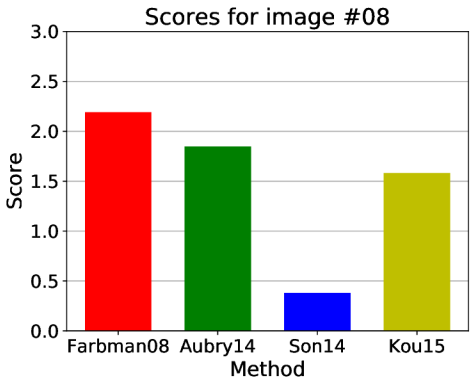

Son14


Kou15

Continuation of questionnaire results on methods' comparison A.1

Original image #08

Scores of methods' results

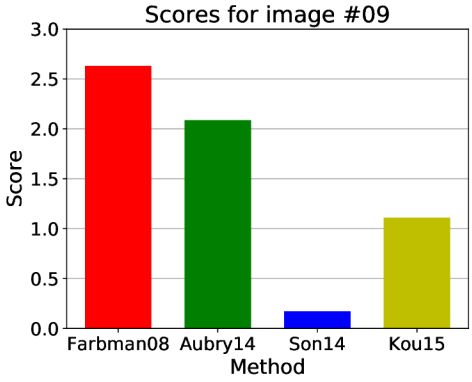





Farbman08 Aubry14 Son14 Kou15

Original image #09

Scores of methods' results



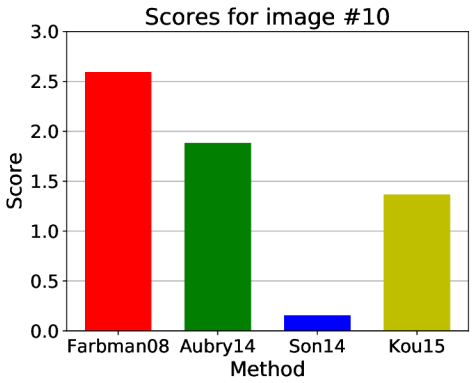



Farbman08 Aubry14 Son14 Kou15

Continuation of questionnaire results on methods' comparison A.1

Original image #10

Scores of methods' results




Farbman08

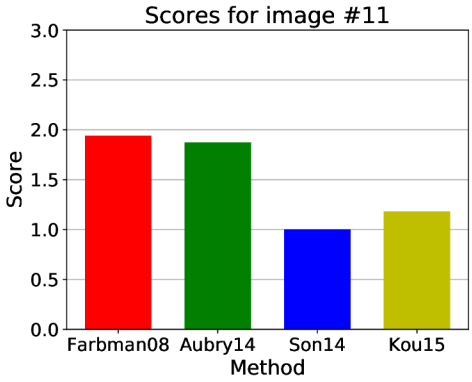

Aubry14


Son14


Kou15

Original image #11

Scores of methods' results




Farbman08


Aubry14

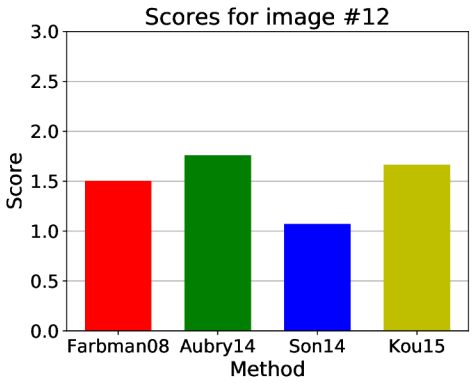

Son14


Kou15

Continuation of questionnaire results on methods' comparison A.1

Original image #12

Scores of methods' results



Farbman08

Aubry14

Son14

Kou15

End of Questionnaire results on methods' comparison

Appendix B

Content of memory media

Memory media contains following content:

- pdf document with this report of thesis
- LaTeXsource codes for this report
- source codes of implemented methods with manual
- binary executables of programs