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

GHOST-FREE HDR VIDEO USING FPGA

GHOST-FREE HDR VIDEO S VYUŽITÍM FPGA

PHD THESIS
DISERTAČNÍ PRÁCE

AUTHOR — **Ing. MARTIN MUSIL**
AUTOR PRÁCE

SUPERVISOR — **Prof. Dr. Ing. PAVEL ZEMČÍK**
ŠKOLITEL

BRNO 2020

Abstract

This thesis proposes an algorithm for multi-exposure ghost-free HDR video acquisition for embedded devices. The Ghost-free HDR acquisition was evaluated on the state-of-the-art FPGA architecture and achieved more than real-time performance of 96FPS on FullHD resolution. The proposed Ghost-free algorithm produces output visually comparable to the state-of-the-art algorithms which are considerably more demanding or not implementable on embedded devices at all.

Abstrakt

Tato práce navrhuje algoritmus pro pořizování ghost-free HDR videa ze sekvence expozic, který je určený pro implementaci ve vestavěných zařízeních. Vlastnosti algoritmu byly ověřeny implementací ve state-of-the-art architektuře HDR kamery, kde je schopen zpracovávat HDR video s potlačením tzv. ghosting efektu rychlostí až 96 snímků za sekundu na FullHD rozlišení, což více než dostačuje pro zpracování v reálném čase. Navrhovaný ghost-free algoritmus produkuje výstup vizuálně srovnatelný s nejmodernějšími algoritmy, které jsou výpočetně řádově složitější a často je nelze na embedded zařízeních ani implementovat.

Keywords

HDR, HDR Acquisition, HDR Deghosting, Embedded Systems, FPGA, Real-time HDR processing

Klíčová slova

HDR, pořizování HDR, HDR Deghosting, vestavěné systémy, FPGA, zpracování HDR v reálném čase

Reference

MUSIL, Martin. *Ghost-free HDR video using FPGA*. Brno, 2020. PhD thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Prof. Dr. Ing. Pavel Zemčík

Ghost-free HDR video using FPGA

Declaration

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana prof. Dr. Ing Pavla Zemčíka. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Martin Musil
August 31, 2020

Acknowledgements

Tímto bych chtěl poděkovat svému vedoucímu prof. Dr. Ing. Pavlu Zemčíkovi za odborné vedení, konzultace a připomínky, které mi pomohly při řešení diplomové práce. Dále bych chtěl poděkovat kolegovi Ing. Svetozáru Noskovi za spolupráci na tvorbě HDR kamery, jejíž architektura byla několikrát publikována v odborných časopisech a na níž byl demonstrován vědecký přínos této práce.

Contents

Chapter 1

Introduction

In the real world, our human vision is capable of seeing and recognising objects in various light conditions, even when they mix in one scene, such as a view from dark room outside to the sunny street. In the contemporary digital world, we are also trying to get this real-looking images into digital form as photography, video etc. One of the current problem in digital image acquisition is very limited dynamic contrast that can be captured from the scene, because the current camera sensors have only limited and linear response to the light, unlike the human eye. This often leads to photos with some white (overexposed) and black (underexposed) sections.

An effort still exists to remove this bottleneck and capture a high dynamic range image (HDR). The first possible way is to assemble a chip with a non-linear response to lightning. They are currently available, but they are still in the early age of development and suffers from some bugs, they have small resolutions, etc. Currently, most spread way how to obtain an HDR image is by merging a sequence of low dynamic range images (LDR) captured by the ordinary camera into one HDR.

The algorithms that merge LDRs into HDR image are known for a quite long time, but they produce a good visual result only with static scenes. In case of any motion, either in the scene or by the camera itself, the ghosting artefacts occur in resulting HDR image. Quite many papers about deghosting techniques were proposed; however, it is still a challenge and a quite open problem, no universal method with reference „deghosted“ result exists.

This dissertation is motivated by a need of many surveillance, security, traffic monitoring systems, and industrial applications that can benefit from HDR video capture. These applications are typically cost-sensitive and so multi-exposure HDR acquisition is often the only feasible option. In these use-cases, the motion in the scene is inevitable and „ghosting“ in such systems, caused by the nature of image acquisition, troubles the applications. Therefore, I decided to develop a method of fast de-ghosting for such applications.

Applications in surveillance, security and industry require high performance in general – we cannot afford slow and demanding offline processing that the best state-of-the-art algorithms require. The essential goal is to capture HDR image fast, to be able to react to a certain situation very fast and or in a given time frame.

Image acquisition systems of this type are still being built on PC based systems; however, this approach is on the decline, since the PCs are expensive, they have large dimensions, and they consume a lot of power. Nowadays, the interest is turning towards compact embedded systems, which are breaking such limits. They often contain low power CPUs accompa-

nied by powerful, task tailored accelerators which require a fraction of power consumption comparing to CPU based systems, while they can deliver even much more performance.

The most efficient circuits are generally considered to be ASICs, which means Application-Specific Integrated Circuits. It is a collective name for single-purpose circuits/accelerators, tailored to provide specific functionality only. However, the manufacturing cost of such circuits is enormous; its manufacturing pays off only with high volumes of chips. The development processes of ASICs are taking place on large FPGAs (Field-programmable Gate Array), which are a completely customisable array of logic gates and registers, which can be interconnected in any desired way; therefore, they offer quite the same flexibility in design as ASICs, but with diametrically lower cost. Nowadays, FPGAs are very popular even in consumer electronics for their computing power, reliability, reprogrammability, low cost, and also low power consumption. These benefits are outweighed by designing time, which is still quite high. Also, not every task is implementable or convenient to accelerate on FPGA.

Some class of image processing algorithms are quite suitable for FPGA acceleration, at least when they uniformly process the image by pixels or blocks. For example, the HDR acquisition, as it was proposed by Debevec and Malik[5] is a typical example of a suitable algorithm. Unfortunately, this algorithm requires static images to produce a good-looking visual output. In case of motion in the scene, the ghost effects appear. As it is summarised later in this dissertation work, deghosting algorithms producing good visual output are very computationally demanding and quite often not even implementable on FPGA. The simpler algorithms are, on the other way, not very successful in deghosting and therefore, they are not suitable for applications in security, traffic monitoring, or industrial applications.

These circumstances led me to set the scientific contribution of this thesis to prove that a multi-exposure ghost-free HDR acquisition algorithm comparable to the state-of-the-art algorithms in quality can be designed for an embedded hardware device and achieves a real-time performance at high resolution.

The dissertation thesis begins with Chapter 2, which contains an overview of state-of-the-art algorithms related to the HDR acquisition and tonemapping. Chapter 2 further contains an overview of state-of-the-art deghosting algorithms, followed by selected deghosting algorithms feasible to be implemented in embedded devices. The thesis continues with Chapter 3 that contains an overview of hardware platforms suitable for implementation of deghosting algorithms, including an overview of embedded system-on-chip solutions. Chapter 3 is further focused on embedded platforms of for HDR acquisition, followed by an overview of existing embedded HDR deghosting solutions.

The proposal of ghost-free merging algorithm, which I developed to fulfil the goal stated in this thesis, is located in Chapter 4, which also contains algorithm evaluation, comparison to related algorithms, and also to the state-of-the-art. The chapter contains an evaluation of performance and power consumption, which demonstrates the engineering contributions of the proposed solution. The chapter ends with an evaluation of scientific contribution and by a summary of possible applications of the proposed algorithm.

Chapter 2

HDR acquisition and deghosting

This chapter contains an overview of state-of-the-art algorithms related to HDR acquisition and tonemapping. The chapter further contains an overview of state-of-the-art deghosting algorithms and also an overview of selected deghosting algorithms feasible to be implemented in embedded devices.

2.1 HDR acquisition

Standard video cameras are unable to capture the dynamic range of visual information the human eye is capable of. The dynamic range is the variation of luminance within a given scene and the obvious goal of image and video acquisition is to capture the whole luminance range of the scene into the captured image. The contemporary sensors are very limited and capable of capturing variations within two or three orders of luminance magnitude, while some scenes contain variations over the five orders. The video cameras are able to select which part of dynamic range is captured and which is lost as under/overexposed, e.g. by a selection of the aperture and shutter speed.

Two main approaches to HDR (High Dynamic Range) image capture exist. First of them is to build special cameras with HDR sensor. Some commercial products start to be available, such as SpheroCam HDR[1], or Panoscan MK3[2] In the academic world, Sakakibara et al. [49] introduced a High-Sensitivity CMOS sensor with gain adaptive column amplifiers and 14 bit analogue-digital converters. Zhao et al. [73] capture HDR using the modulo camera. All the above approaches require the availability of special HDR sensors or generally expensive and technologically demanding equipment. Regarding the HDR sensors, it is questionable whether some physical limit in a dynamic range will eventually be reached and what it will be.

The second and more frequently used approach is based on standard sensors/cameras which captures the high luminance range in the scene sequentially, by the acquisition of multiple images typically with varying exposure times [5, 37, 48, 34]; such sequence is then merged into one HDR image. The individual images can be captured simultaneously, e.g. using a beam splitter with several CCD/CMOS sensors [58], or, more often are gathered sequentially using a single image sensor which causes ghost effects by a motion of objects during the sequence acquisition. This approach is technologically less demanding and results in cheaper systems.

[1] https://www.spheron.com/
[2] http://www.panoscan.com/

HDR acquisition algorithms

Two main approaches how to merge differently exposed standard images into an HDR image exist, the first and more efficient approach involves a combination of pixels in the image domain (direct merging of pixels). As an example, a method presented by Mertens et al. [34] combines multiple exposures directly without any knowledge of the camera response function(CRF). In this approach, only the best parts of frames from each exposure are exploited. A resulting HDR image is obtained as a weighted average of pixel values across the exposures:

$$I_C = \sum_{k=1}^{N} w\left(Z_k\right) Z_k \tag{2.1}$$

where I_C is a composite image, Z_k is a pixel value and $w\left(Z_k\right)$ is a weight of a pixel. This approach produces the HDR images which can be directly displayed on LDR (Low Dynamic Range) monitors.

The second approach is based on merging in the radiance domain, in the meaning of real illumination in the given scene. Algorithms using this approach are attempting to calculate the exact value of luminance in the scene. These methods require knowledge of the camera response function [5, 48, 37], which is the response function of the camera sensor to the incident light (see Section 2.1). The inverse function of CRF is then applied to obtain an image with approximately linear response to light. The CCD and CMOS technology generally do have a linear response function, but the image results are often affected by postprocessing algorithms, for example, by gamma-correction or by white balance. In general, RAW images are preferable for HDR composition because they contain data obtained directly from CCD/CMOS sensors without any postprocessing, and therefore it can be assumed that they have a linear response function. Unlike the merging in the image domain, this class of algorithms produces an image with higher bit-depth, which is not directly displayable on standard LDR devices. The HDR images have to be post-processed by algorithms commonly called tone mapping operators. The operators reduce the bit-depth of the HDR image while they preserve all important image details.

The first and most straightforward approach is to select the pixels from the longest but still unsaturated exposure. The resulting pixel value in the HDR image is calculated according to equation:

$$L_p = \frac{Z_{longest_p}}{t_{longest}} \tag{2.2}$$

where Lp is the resulting pixel value p from HDR image, the pixel value $Z_{longest_p}$ is the value of the pixel from image with longest exposure time where pixel is not saturated and $t_{longest}$ is the exposition time of this image.

Debevec and Malik [5] proposed an algorithm which can fuse multiple photographs into a high dynamic range radiance map whose pixel values are proportional to the true radiance values in the scene. The contribution of each pixel is determined from the weight function shown in Figure 2.1. Resulting pixel value p in HDR image is calculated as a weighted average of each pixel exposures:

$$L_p = \frac{\sum_{i=0}^{N} w(Z_{ip}) \frac{Z_{ip}}{t_i}}{\sum_{i=0}^{N} w(Z_{ip})} \tag{2.3}$$

where L_p is the resulting pixel value p in HDR image, N is the number of input images, Z_{ip} is the value of a pixel p in image number i, t_i is the the exposure time of image i.

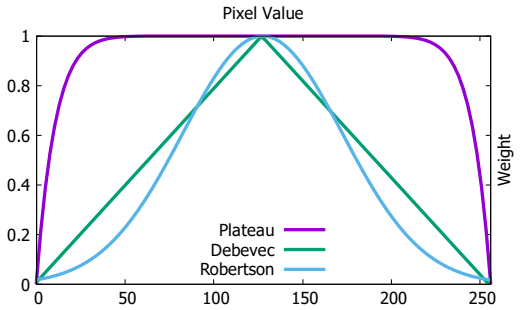


Figure 2.1: Pixel weighting functions proposed by Debevec [5], Robertson [48] and plateau weighting function (from Luminance HDR software[3]).

This algorithm can use various weighting functions. The weight function by Robertson (shown in Figure 2.1) is a Gaussian-like weight function. In the HDR processing applications (e.g. Luminance HDR[3]) the "plateau" function is often used (Figure 2.1) which it is defined as follows:

$$w_z = 1 - (2z - 1)^{12} \tag{2.4}$$

where z is a pixel value. In case of image captured with linear sensor, there is no need of assigning variable weights to pixels in the linear range, except the extreme pixel values, where the pixel value could be distorted.

Obtaining camera response function

The camera response curve important information for HDR image creation. CRF is a curve which indicates the conversion relationship between the brightness of the scene and the resulting values of the recorded image. This response is most influenced by the image post-processing in the camera. It's important to work around this process since composing of HDR images require the values with a linear transfer characteristic between the light in the scene and the values in the image.

Analog data from the CCD/CMOS sensor are converted to digital using A/D converters and then further processed in the camera. Range of adjustments varies by camera and its settings. Common modifications include remapping of pixel values (change of contrast and brightness, gamma mapping curve, etc.), colour correction (e.g. increased colour saturation), noise reduction, sharpening, and more. Some of the newer consumer cameras, for example, also carry out the reduction of the optical lens distortion. It is possible to get around these corrections by using the RAW format, supported by some cameras, where the data from the sensor are stored unchanged. RAW output is almost linear, only with a few exceptions:

- **The level of black** - Because of the chip design, charge amplifiers, A/D converters and their noise, the level of black can be shifted from zero upwards. This means that black has a higher value than zero. To eliminate this effect, we have to determine this value and subtract it from the image data.

- **Quantization** - Error in quantization arises during the conversion from analogue to digital values. The converter with a higher number of bits (10, 12, ...) can reduce the absolute error value. Quantization also causes higher inaccuracies in the dark

[3] http://qtpfsgui.sourceforge.net/

values of the image (during the short exposures, the higher range is mapped into a few values near to zero).

- **Saturation** - Photodiodes on camera sensor have only limited charge capacity, which determines the level of saturation. Above the level of saturation, the camera gives the same numerical response for all values of the input luminance. These values are not applicable for HDR image composing and therefore have to be suppressed, e.g. by using a weighting function.

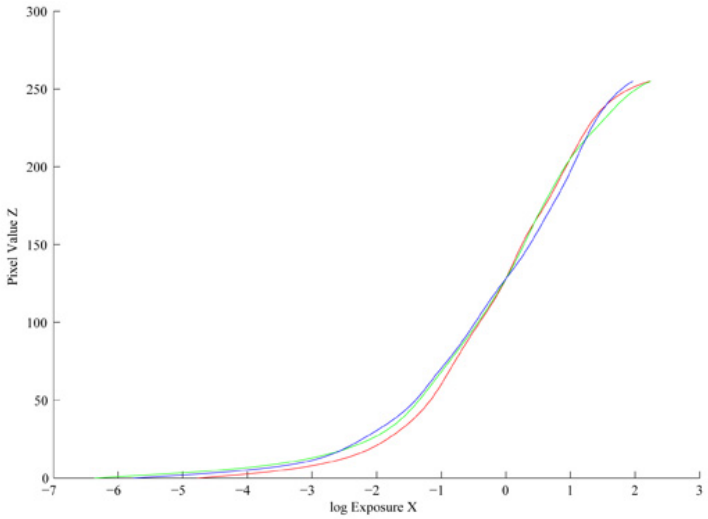


Figure 2.2: Response curve of the Canon EOS 350D (retrieved from http://cybertron.cg.tu-berlin.de/eitz/hdr/).

Several algorithms for calculation/estimation of response function exist if we don't have an opportunity to obtain the picture with a linear response to incident light. The basic method is using a standardized table composed of several fields in grayscale which are photographed and then compared with the camera output. The response function is created from the differences. However, the measurement is complicated because the table must be uniformly illuminated and shall not shine.

Algorithm by Debevec

This algorithm is described in the paper *Recovering High Dynamic Range Radiance Maps from Photographs* [5]. The initial assumption is that the scene is static and the shots are taken so quickly that we can ignore changes in scene illumination. Under these circumstances, we can assume that the intensity of illumination E_i is constant for each pixel. The values of each pixel will be marked Z_{ij} where $i = 1, ..., N$ is a one-dimensional index specifying the position of the pixel in the image and $j = 1, ..., P$ is the index across the different exposure times Δt_j. The relationship that exists between the E_i and Z_{ij} is defined as:

$$Z_{ij} = f(Ei * \Delta t_j) \tag{2.5}$$

where f is an unknown camera response function which we assume to be monotonous.

Lets define a function g as the natural logarithm inverse to the function: $g = lnf^{-1}$. We get an equation in the form:

$$g(Z_{ij}) = lnE_i + \Delta t_j \tag{2.6}$$

The values of Z_{ij} and Δt_j are known, lighting E_i and function g are unknown. The advantage is that searching the function g implies the searching of finite number of values of $g(z)$, where $z =< z_{min}; z_{max} >$ is a finite set of values that pixels can take. The problem is then reduced to a search of finite number of values of $g(z)$ and N values of lnE_i, which are minimizing the value of the following quadratic optimization function:

$$\mathcal{O} = \sum_{i=1}^{N}\sum_{j=1}^{P}[g(Z_{ij}) - \ln E_i - \ln \Delta t_j]^2 + \lambda \sum_{z=Z_{min}+1}^{Z_{max}-1} g''(z)^2$$

Other algorithms

The method presented by Robertson [48] does not put any restrictions on the shape of the resulting response function. This method assumes a Gaussian weight function. Using the Gauss-Siedel iteration, the authors seek the solutions of the objective function which they defined. The method presented by Mitsunaga [37] approximates the camera response function by a polynomial of N-th degree. The authors are looking for coefficients of the polynomial by minimizing the error function, which they defined. The advantages of the method include the ability to determine the exact ratios of exposures. Also, many other methods exist, such as the histogram-based method or a method attempting to derive the response from a single image.

2.2 HDR tone mapping

HDR acquisition algorithms produce images which are not directly displayable by current display technologies. The dynamic range of the HDR image has to be compressed to be able to display such an image. Such a process is commonly called tone-mapping. Application of tone-mapping should compress only the range of values; however, the visual information should be preserved – this ability strongly depends on individual algorithms and their properties. This dissertation addresses tone-mapping only marginally; still, a short overview is convenient for the coherence of the topic.

Displaying the HDR content is still a challenging topic, as standard displaying devices are able to represent only a limited dynamic range, typically 8 or 10 bits per channel. To display an image with higher bit-depth, the HDR images have to be post-processed by algorithms commonly called Tone Mapping Operator (TMO) which reduce the bit-depth of HDR image so it can be displayed using standard devices while preserving all important details.

In general, two main approaches to displaying HDR content exist. The first approach is to use specialized HDR monitors that directly render the HDR content; such displays still have some limitations, they are expensive, and they often should be used in a very controlled environment. Thew second approach is based on the application of dynamic range scaling with an effort to reduce the dynamic range but to preserve local contrast in the scene details. This process, as mentioned above, is called Tonemapping (or applying

Tonemapping operators). Many operators exist [9, 47, 11, 6, 31, 7] and they are divided into two main categories - global and local operators.

Global operators use the same mapping function for all pixels of the image. Parametrization of the function depends on global image characteristics, such as average, minimum, and maximum values of luminance.

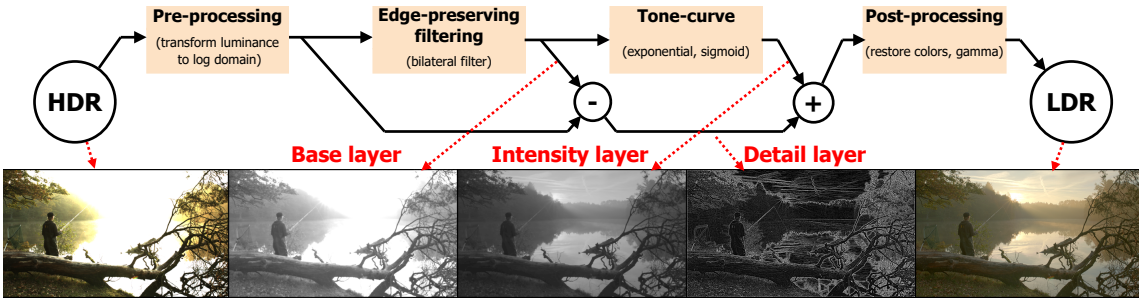


Figure 2.3: Illustration and principled scheme of local tone mapping algorithms (Image retrieved from dataset by Froehlich et al. [13]).

Local operators generally preserve more details than the global ones as they use information from neighbouring pixels to estimate local illumination and thus adapt compression to local luminance conditions. The most frequently used approach is separation of HDR image into *base* and *detail* layers (see Figure 2.3). The *base* layer contains large scale variations of luminance and the *detail* layer contains local differences, which holds the details of the scene. The base layer can be obtained from luminance in the logarithmic domain by using low-pass edge-aware filtering, e.g. by Bilateral filter (BF), as proposed by Durand [9], Edge-avoiding wavelets [10], or by estimation from gradient domain (Gaussian pyramids [11]). The *base* layer, which is responsible for high dynamic range, can be compressed because fine details are preserved in *detail* layer.

2.2.1 HDR compression

High Dynamic Range (HDR) imaging technologies can provide high levels of immersion through a dynamic range that meets and even exceeds the instantaneous range of the Human Visual System (HVS). This increase in the level of immersion comes at the cost of significantly higher bit-rate requirements compared to those associated with conventional imaging technologies. As a result, efficient HDR-relevant coding solutions have to be developed.

Backwards-compatible HDR compression methods are designed so that legacy decoders, which can manage only Low Dynamic Range (LDR) images, are still able to decode and display a tone-mapped version of the HDR image/video. HDR-capable decoders, if available, would be able to decode the full stream and deliver the HDR image experience.

Figure 2.4 shows a block diagram of backward-compatible HDR image and video encoders. The base layer encodes a tone-mapped 8-bit LDR representation of the HDR input using a fully compatible legacy encoder and decoder. The enhancement layer contains the difference (residual) between the inverse tone-mapped base layer and the original HDR input, which is used for reconstructing of HDR content in the HDR devices.

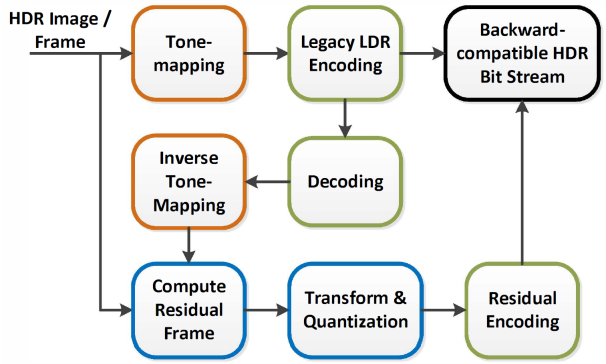


Figure 2.4: General structure of backward-compatible HDR image and video encoders (retrieved from [72]).

Backward compatible image compression

At the first place, several methods for encoding HDR images were proposed. In 2004,

Ward and Simmons[65] proposed a backwards-compatible HDR extension for JPEG, whereby the HDR image is tonemapped to an 8-bit LDR image which is then encoded by a legacy JPEG encoder. The ratio image between the original HDR image and the tone-mapped version is down-sampled and stored as a tag in the header file. This ratio image can be used by HDR-capable decoders to reconstruct the HDR content, while all other legacy devices would simply ignore the tag and directly display the tone-mapped LDR image.

Spaulding et al. [53] proposed layered coding for JPEG gamut extension. In the base layer, an image with a clipped colour gamut is encoded. In the enhancement layer, a residual image is formed in a sub-band. This residual image is defined as the arithmetic difference between an input ERIMM RGB colour space image and the encoded sRGB foreground image (limited to 8 bits). The main advantage of the approach proposed in [53] is that the format is backwards-compatible with existing JPEG image codecs.

Backward compatible video compression

Mantiuk et al. [32] were the first to propose layered coding for backwards-compatible HDR video compression. Their method was designed as an extension to the MPEG-4 compression standard. The authors introduce a colour space transformation that facilitates comparisons between LDR and HDR pixels. A reconstruction function is then proposed which predicts the value of an HDR pixel based on the value of the corresponding LDR pixel. A non-linear function is used for encoding the HDR residual information. This is then added as side information to the bitstream and can be used by HDR-capable decoders to obtain the full range of visible luminance values. In order to facilitate a smooth transition from LDR to HDR in the decoder, tone-mapping and residual video stream calculation are performed as a preprocessing step before encoding. The novelty of this method is that it employs an advanced Human Visual System (HVS) model to achieve better compression performance. The HVS model selectively pre-filters the residual stream in order to remove imperceptible high spatial frequency information, thus reducing its bit-rate requirements after MPEG-encoding.

Mai et al. [74] proposed a backwards-compatible method that aims to find an optimal tone-curve for mapping the input HDR image/video to a backwards-compatible 8-bit LDR image/video, which can then be compressed by a conventional video codec such as H.264. The reconstructed image/video can then be displayed on a conventional LDR display or can be inverse-tone mapped and augmented by an optional enhancement layer containing an HDR residual signal (also compressed by the codec) for display on an HDR display. The tone-curve optimization aims to minimize the quality loss due to tone-mapping, encoding, decoding and inverse tone-mapping of the original image/video.

2.3 HDR deghosting

The HDR merging algorithms [5, 37, 48, 34] summarized in Section 2.1 are suitable for static scenes only. Motion of objects during the image sequence capture causes adverse effects called *ghosting*. To reduce such effects, various methods to detect and remove ghosting from HDR images have been developed.

Figure 2.5: Image on the left includes ghosting artifacts, that has to be removed or reconstructed (right). Retrieved from http://www.flickr.com/photos/nuwomb/.

The problem of removing motion artefacts for sequential HDR imaging has been the subject of extensive research and has led to two major type of approaches. The first type assumes that the images are mostly static and that only a small part of the scene contains motion. These de-ghosting algorithms use the input frames to determine whether a given pixel is static or has motion and then apply different merging algorithms in each case. For static pixels, the traditional HDR merge can be used. For motion pixels, many algorithms use only a subset of exposures (in many cases only one) to produce a deghosted HDR. The fundamental problem with these techniques is that they cannot handle scenes with large motion if the moving parts of the scene contain HDR content.

The second type of approaches attempts to align the input sources to a reference exposure before merging them into an HDR image. The most successful algorithms use optical flow to register the images, but even these methods are still brittle in cases of large motion or complex occlusion/dis-occlusion. Since the aligned images produced by these algorithms often do not align to the reference very well, the resulting HDR still contain the ghosting artefacts. For this purpose, the alignment algorithms for HDR often introduce special merging functions that reject the information from aligned exposures in locations where they do not match the reference. In such a case, the HDR content in these regions are not fully reconstructed.

According to the goal of this dissertation, I focused on algorithms feasible of capturing ghost-free HDR images in real-time. Thus I further reviewed mostly simple, computationally unpretentious methods, that could claim the real-time performance. I reviewed mostly the methods categorised by Tursun [59], Srikantha [54] and other authors as „motion object selection“ methods. Anyway, a short introduction into the demanding optical flow and patch-based algorithms is presented.

2.4 Motion object selection methods

This dissertation work focuses on embedded systems and real-time processing; therefore, only simple, computationally unpretentious methods, categorised by Tursun [59], Srikantha [54] and other authors as „motion object selection“ methods are reviewed in this subsection. The optical flow-based and patch-based algorithms are, due to their high computational demands, reviewed only for the coherence of the topic. Also, the global image registration is not addressed, as we assume only static cameras.

In the work of Sidibe et al. [51], the ghosting regions are detected based on the observation that the order relation is given by Equation 2.7 is satisfied for pixels which remain static between two images, and can be broke down for motion pixels. Therefore, they detect possible ghosting regions by checking the order relation between k consecutive images, and by marking pixels for which the relation breaks down at least once.

$$if \Delta t_i > \Delta t_j, \;\; then \;\; L_i > L_j. \tag{2.7}$$

Where Δti and Δtj are exposure time of images L_i and L_j. This method does not verify the real increment of a pixel value which causes the ghost detection to fail relatively often. The order relation only works if the pixel is not under- or over-exposed. For instance, a white pixel in a shorter exposure will remain white in a longer exposure, and a black pixel in a longer exposure remains black in a shorter one. Therefore, the authors discard under and over-exposed pixels when checking the order relation between consecutive images. Concretely, they exclude pixels which are outside the range [20,. . . , 240] (for 8-bit pixel values). [51]

Once ghost regions are detected, artefacts-free HDR can be created. For all pixels outside a ghosting region, HDR generation proceeds in a conventional manner, i.e. the pixel value in the HDR is a weighted average of the corresponding pixels in the differently exposed images, as proposed Debevec and Malik [5]. For a pixel inside a detected ghosting region, a common approach is to substitute the pixel value by the corresponding value in the best exposure image for that region. For each region, the best exposure is chosen as the one with the lowest number of under-or over-exposed pixels. Sidibe's method gives

goods results in some cases, it, unfortunately, reduces the dynamic range of the HDR by considering only one exposure. [51]

Sidibe et al [51] claimed that according to experiments on various sequences, the order relation-based method and the predicted colour method give more precise results than the variance-based method. However, the measure of variance works well only if the colour of the moving object is clearly distinguishable from the background.

Kao et al.[22] processes two images with $\pm 1EV$ difference in exposition times. Since they know exposition time, the following relation between pixel values is expected:

$$L_2(p)/L_1(p) = 2, \Delta t_2 = 2 * \Delta t_1 \tag{2.8}$$

If this relation is not fulfilled, the pixel is marked as a ghost and is omitted from the merging algorithm. The saturated pixels, which are under or over a fixed threshold, are omitted from the ghost detection process too. Kao et al. uses the motion estimation for aligning the source images, which works over macroblocks of 16×16 pixels.

Gallo et al. [14] assumes a linear dependency between couples of pixels when they „see“ the same radiance levels, based on knowledge of exposure times. The following relation between the images is expected:

$$L_i = L_j \cdot \frac{t_i}{t_j} \tag{2.9}$$

Any image spot violating this linear relation is considered as containing a motion. All images are registered to the reference image L_{ref}; to suggest a good reference frame, they find the saturated pixels in each image of the stack, then they remove small saturated regions with morphological operators (erosion followed by dilation) because such area's neighbourhood usually contains enough information to avoid artefacts. Finally, they pick the exposure with the fewest remaining saturated pixels. [14]

The reciprocity assumption states that if the radiance of the scene does not change, the exposure time and the irradiance are linearly related through the exposure time Δt:

$$X = E \cdot \Delta t \tag{2.10}$$

Aside from over and under-saturated pixels, Equation 2.10 should only be violated when the scene changes. Therefore, the equation could be used to decide if the irradiance at a given pixel in the reference frame can be combined with the corresponding pixel in another image in the stack. In practice, however, a small misalignment or imprecise estimation of the camera response function can produce large deviations from this behaviour . [14] To increase a robustness and prevent rising of such artifacts, the algorithm operates on relatively large rectangular patches (e.g. 40x40 pixels) rather than individual pixels. Patches with a large number of not corresponding pixels are omitted from merging, causing visible artefacts to occur at their boundaries; Gallo et al. [14] suggest their suppression by Poisson blending.

Raman et al. [46] extended the work of Gallo et al. [14] so that it does not require any knowledge of the CRF or exposure settings. They introduced an intensity mapping function (IMF) obtained from the static part of the scene – they assume that upper 5-10 image lines are usually static. The authors assume the motion is mostly confined to the ground plane of the scene. This assumption may be very limiting, and it can work only for certain scene compositions.

Grosch [15] proposed a simple method based on the estimation of pixel value from the known exposure time and CRF. In opposite to the most of the algorithms that require a static scene and direct correspondence of pixels to obtain a CRF, Grosch uses the algorithm presented by Grossberg and Nayar [16] to recover a CRF from a non-aligned sequence with object motion. This algorithm calculates the response function based on cumulative histograms and is mostly unaffected by camera or object motion. [15]

With a known camera response function, they can predict the pixel colour from one image to another. For each pair of consecutive images, they test if the real colour in the second image is well approximated with the predicted colour from the first one. If the pixels at the certain position do not fit the estimation, the corresponding region is marked as ghosted into the *error map*. [15] To increase the robustness and eliminate the influence of the noise in the source images, the author uses a user-defined threshold for the pixel colour comparison.

Wu et al. [66] algorithm estimate the CRF from regions where RGB vectors remain fixed with respect to the changes of exposure. The algorithm refines motion detection by a combination of pixel order relation from Sidibe et al. [51] and pixel value estimation from Grosch [15].

Wang et al. [63] proposed the motion region detection method, that is motivated by the inter-frame difference method for video sequence that does subtraction to compute the difference between adjacent frames on the intensity domain. To enable it, the algorithm normalises all images L_i according to the reference image L_{ref}. For each pixel, if the corresponding difference value is bigger than a certain threshold, then the pixel is considered to be in a motion region. This method is commonly used on motion detection of video stream. [63]

The threshold value is determined from median pixel value in each image and is adaptive to avoid certain artefacts – the threshold for the under and over-exposed areas is increasing, because the brightness changes only little in spite of adjusting the exposure levels because it has reached saturation. The bitmaps with detected motion are further strongly refined using morphological operations. The tolerance ratio should also be adjusted by the user to provide the best visual result. [63]

The algorithm of Jacobs et al. [18] is calculating pixel variance over the exposures to detect the presence of motion. The *Variance Image* is created, storing pixel's variance over the exposures in a matrix with the same resolution as input images. Further, they ignore under and over-saturated pixels in *Variance Image*. The *Variance Image* is transformed into binary map (equivalent of *ghostmap*, see on Figure 2.6)), with movement clusters, which are formed by comparing the *Variance Image* with fixed threshold. The *Variance Image* is supplied by *Uncertainty Image*, which is calculated using the local variance, obtained from a histogram of a small 2D window; 5x5 pixels in size [18].

For each movement cluster, they substitute the irradiance values with the irradiance values from only one image. Substituting an entire region with irradiance values from one image introduces artefacts at that region's borders. To reduce these artefacts, they substitute pixel values with *Variance* and *Uncertainty Image* entry above a certain threshold (higher than the threshold used at the movement clusters) with a weighted average of the reference image and the irradiance value from in the selected image. [18]

Pece et al. [44] algorithm extracts median threshold bitmap (MTB [64], see Figure 2.7) from each of the input images. Any difference between the threshold maps of input images and the reference image, presented typically by the mid-exposure one, is marked as a motion-region. To remove the effects of the noise, the motion map is refined using mor-

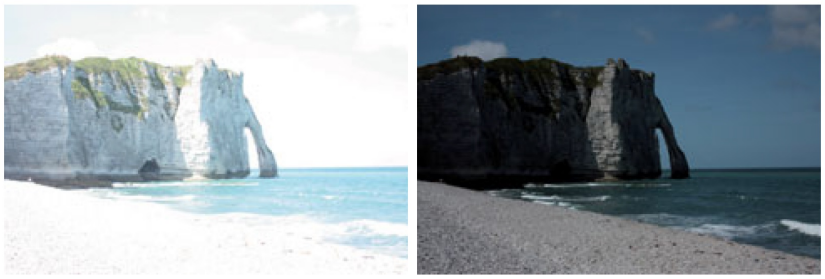
Figure 2.6: The Figure shows the results of variance based deghosting method by Jacobs et al. [18]. The variance map (bottom left) is obtained from the image sequence (upper row) and used to generate the HDR image (bottom right). Figure obtained from [18].

phological operations such as erosion and dilation. The pixels in the motion-regions are assigned smaller weights during the HDR construction. This method is strongly dependent on scene composition since its reliability is strongly dependent on the median image value. Additionally, the pixels with a value close to the threshold may be falsely detected as ghosts.

Min et al. [35] improved method of Pece et al. [44] and introduced multi-level threshold map, where thresholds are selected to divide the image into multiple regions according to the pixel intensity, each region having the same number of pixels (see Figure 2.8). Any difference between the threshold maps of input images and the reference image, presented typically by the mid-exposure one, is marked as a motion-region. Introduction of multiple histogram regions, in opposite to Pece et al. [44], allows for the incorporation of a tolerance in which shifts of pixels within neighbouring regions are not evaluated as motion. The algorithm suffers from dependence on scene composition and image histogram layout. Min et al. [35] further improved the algorithm in a follow-up article [36] by employing a noise reduction phase, which incorporates an additional set of rules for spatially neighbouring pixels. Unfortunately, the algorithm needs to use a large, performance and memory demanding, spatio-temporal smoothing filter. The above methods by Pece et al. [44] and Min et al. [35, 36] are using coarse morphological operators, such as erosion and dilatation, to suppress false detection rising on edges or by noise.

An et al. [1] proposed a method for multi-exposure fusion without the ghosting effect. The method evaluates the photometric relation of images from sequence to the reference image, producing binary ghostmaps. This ghostmaps are refined using the ZNCC (Zero Mean Normalized Cross-Correlation), which evaluates the similarity of the ghostmap patterns. By default, they use an 11×11 patch for photo-metric relation test and a 33×33 patch for ZNCC calculation.

Moon et al. [38] published a short article where they proposed HDR fusion method with ghost-free effect. The author claims that the method is intended for embedded devices and



(a) Original Exposures: +2 and -2 stops respectively.

(b) Bitmaps generated by MTB.

Figure 2.7: The Figure shows the two images and corresponding Median Threshold Bitmaps (MTB), proposed by Pece et al. [44]. Figure obtained from [44].

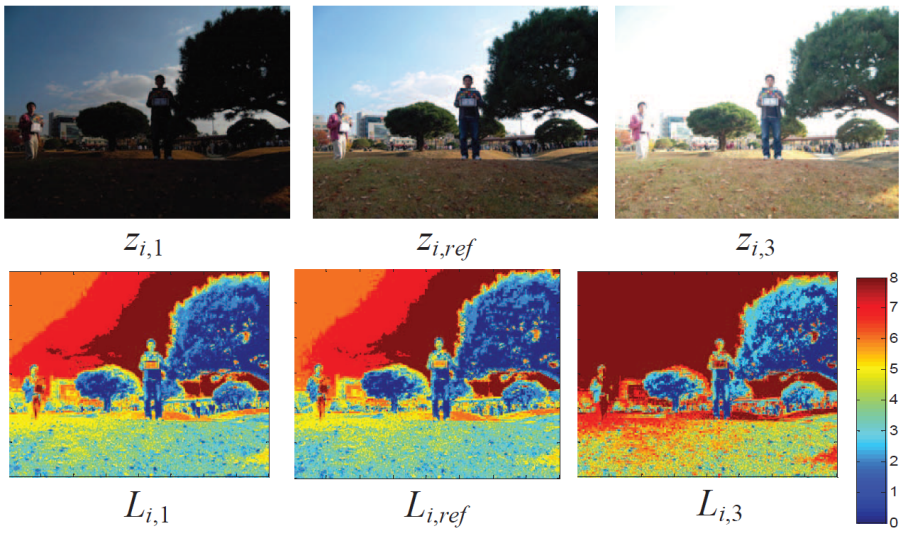


Figure 2.8: The figure presents the intermediate step of deghosting algorithm by Min et al. [35]. The source sequence is on the top, the bottom images shows the multi-level threshold maps for corresponding images on the top. Figure obtained from [35].

fast processing, since it does not require the demanding morphological operations, required by many other algorithms [44, 35, 1] for ghostmaps refinement. They introduced a simpler non-ghostness probability, which is combined with a conventional fusion weight [34] to yield a ghost-free fusion weight. All of the images are photometrically calibrated toward the reference image by an image transform from the result of histogram matching between

the reference image and their individual exposure images. Unlike in their previous work [1], they use „soft“ assigned weights instead of binary ghostmaps.

Srikantha et al. [54] propose a method which works on input images with linear CRF. Their work is based on the assumption that if the pixels from different exposures capture a static region of the scene, they must be linearly dependent since they are equal to the multiplication of sensor irradiance and exposure time. The pixels which do not follow the linearity and potentially cause ghosting are found using singular value decomposition (SVD) of a matrix containing pixel intensities from all exposures. This matrix is reconstructed using only the largest singular values, forcing the linearity between the corresponding pixel intensities of different exposures. The reconstructed pixel intensities are used to produce a ghost-free HDR image. [59]

Bouderbane et al. [3] implemented simple ghost removing algorithm on FPGA based platform. They were inspired by the work of Sidibe et al. [51] and presented the algorithm based on the modification of Debevec [5] weighting function. The idea of the methods is to adjust pixel weights based on the deviation from the reference image [51]. The function gives a higher weight for pixels whose value are closed to the reference value and low weight for pixels whose value diverges considerably from a reference value. Consequently, they achieved the same performance as the Debevec and Malik [5] standard algorithm with a ghost removing in a radiance domain, right before HDR data generation. [3]

2.5 Motion object registration methods

The following algorithms are not suitable for real-time processing; however, I reviewed them for the coherence of the HDR deghosting topic and also because they are part of the state-of-the-art in terms of deghosting quality. Achieving good visual results comparing to such algorithms is also one of my side-goals.

Patch-based algorithms

These approaches attempt to align the different LDR exposures before merging them into the final HDR image. Although the alignment of images has long been studied in image processing and vision communities (e.g. Zitová and Flusser [76]), its application to HDR imaging has special considerations. Here, the input images are not of equal exposure, so the colour constancy assumption of many algorithms is violated. Even if we map images to the same radiance space using the camera response curve (Debevec [5], Mitsunaga [37]), they will have regions that are too dark/light and therefore invalid during the alignment. This makes standard image registration techniques unsuitable for this application. [50]

The quality of the HDR images produced by these techniques is fundamentally limited by the accuracy of the alignment. Even the state-of-the-art optical flow algorithms are brittle in cases with complex motion and occlusions, which is why many use special HDR merging steps to reject misaligned images (as in deghosting) and cannot use standard merging techniques. Furthermore, optical flow cannot typically synthesise new content and thus cannot handle disoccluded content that could be made visible when aligning one image to another. [50]

The algorithm proposed by Sen et al. [50] is a patch-based energy minimisation formula. The algorithm produces an HDR image from a set of LDR images captured with different exposures which is aligned to the reference image L_{ref} and which is also an LDR image that

contains the best-exposed pixels. The resulting HDR image contains as much information as possible from the well-exposed pixels from the L_{ref} image (see Figure 2.9). In places where L_{ref} is not well exposed, every patch in the image H at a given exposure should have a similar patch in one of the LDR images after exposure adjustment (coherence). Also, every exposure adjusted patch in all L_k images should be contained in H at exposure k (completeness). The iterative approach performs joint optimisation of image alignment and HDR merge process until all the exposures are correctly aligned to the reference exposure, and a good quality HDR result is produced.

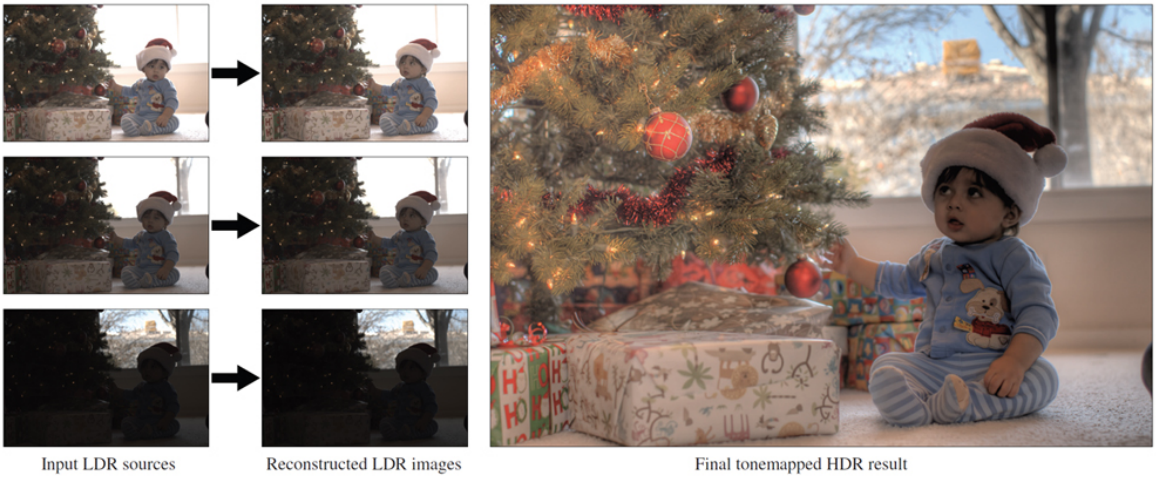


Figure 2.9: The figure shows the source sequence, images reconstructed by patch-based algorithm by Sen et al. [50] and the resulting HDR image. Image obtained from [50].

Orozco et al. [43] presents a method which consists of both ghost detection and image registration steps. In the ghost detection step, the detection algorithms of Pece [44], Jacobs et al. [18], Sidibe et al. [51] and Grossberg et al. [16] are compared, and it was found that the IMF based ghost detection of Grossberg et al. is the most accurate. In the image registration phase, an intensity-based method without feature detection is employed. The image with the best exposure is selected as the reference image. A bounding box is fitted around the previously detected motion regions. Next, the region in each bounding box is registered by translation and rotation to the reference image. The Sum of Squared Distances (SSD), Normalized CrossCorrelation (NCC), Mutual Information (MI) and MedianBitmap Difference (MBD) are compared as a similarity measure for the registration. The authors state that NCC has the best computational cost and performance. In order to speed up the process, the registration is performed using the pyramid structure of the images, from coarse to fine resolution. However, since the registration applies only translational and rotational transformations, more complex motions caused by objects with deformable bodies are not handled. [59]

Hu et al.'s more recent work [17] proposes a PatchMatch[2] based HDR reconstruction algorithm with energy minimization (see Figure 2.10). Among the input LDR images, the one with the largest number of well-exposed pixels is selected as L_{ref}. In the next step, for each input LDR image L_i a latent image T_i is synthesised. Latent images are similar to L_{ref} where it is well-exposed. In under- or over-exposed regions, a matching patch is found using the PatchMatch algorithm in other input images. Using the matching patches and the intensity mapping function obtained with the histogram-based method of Grossberg

Figure 2.10: The figure shows the source sequence (left column), images reconstructed by algorithm by Hu et al. [17] (middle column) and the resulting HDR image (right). Image obtained from [17].

and Nayar[16], the latent images are obtained by minimising the following energy function:

$$\varepsilon(T, \tau, u) = C_r(T, L_{ref}, \tau) + C_t(L, T, u) \tag{2.11}$$

where L, T and u are the sets of input images, latent images and coordinate mappings to matching patches, respectively. The C_r and C_t terms measure the radiometric and the texture consistencies between the reference image and the input images, respectively. As opposed to Sen et al. [50], Hu et al. [17] does not require the CRFs of the input images to be linear. In certain comparison studies, it is observed that Hu et al. [17] was more successful at producing noise-free outputs whereas Sen et al. [50] was better at preserving texture details. [59]

Optical flow-based algorithms

The approaches in this group are mostly based on optical-flow estimation, which is a well-studied problem, especially in stereo vision applications. In the HDR domain, optical-flow estimation must also take the exposure differences between the input images into account. The accuracy of the estimation is very critical for the quality of the outputs since any mismatch results in undesirable artefacts. In addition, the use of optical-flow presents other challenges such as handling the occlusion, noise, or large displacements in the scene. [59]

Zimmer et al. [75] use state-of-the-art optical flow approach to register LDR exposures before the merging process. They minimise their proposed energy function that uses a data term and smoothness term to reconstruct saturated and occluded areas. After alignment, the displacement fields obtained with subpixel precision are used to produce a super-resolved HDR image. The main advantage of the proposed strategy is that the resulting dense displacement fields can describe arbitrary complex motion patterns, which

is indispensable when dealing with complex camera motions or motion objects in the scene. Another attractive aspect is that they do not require knowledge of the camera response curve or the exposure times. Concerning efficiency, they were able to achieve reasonable run times on sequential CPU architectures, whereas parallel GPU implementations reduce the computation times to a few seconds.

Ferradans et al. [12] find dense correspondence of input images in the radiance domain with respect to the reference image. In order to detect the mismatches in the estimated flow fields, the input images are warped using the estimated fields, and the absolute difference map of each pixel is calculated. Instead of applying a fixed threshold to the difference map, its histogram is modelled as a mixture of Gaussians. The pixel intensities corresponding to the flow vectors causing the mismatch are assigned zero weight in HDR reconstruction. The information from the remaining pixels in each input image is fused in the gradient domain. Jinno and Okuda [19] use a novel weighting function which has significantly smaller overlap between the contribution of input LDR images to the radiance domain. The proposed method assumes that the global alignment is already performed. Displacement, occlusion, and saturation regions are modelled as Markov Random Fields. The optimal parameters are found by minimising the energy function (see [19]). [59]

CNN based algorithms

The latest published algorithms are based on popular Convolution Neural Networks (CNN). Kalantari et al. [20] based their approach on optical flow from Liu et al. [28] and merges images into HDR using CNN. At first step, the source images are normalised to the same level of luminance as the reference (middle) image – similarly to Wang et al. [63] and many others. Then, the optical flow algorithm of Liu et al. [28] is used to align the images. Such aligned set is merged using CNN network trained on their dataset containing ground truth sequences. The CNN is responsible for removing the ghosting artefacts appearing on the edges of motion regions. Yan et al. [69] proposed a similar approach; however, their proposed CNN uses not only surrounding information of a pixel as Kalantari et al. [20], but also considers the information from other frames.

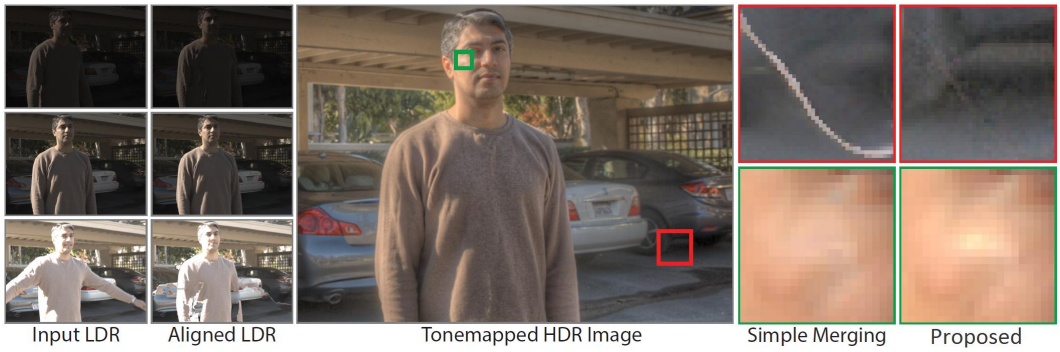


Figure 2.11: The figure presents the results achieved by Kalantari et al. [20]. From the left – the source sequence, images aligned by optical flow by Liu et al [28], resulting tonemapped images and the details of marked region merged by „simple“ merging (probably by Debevec and Malik [5]) and by proposed CNN based method by Kalantari et al. [20].

Chapter 3

Embedded HDR acquisition and deghosting

This chapter contains an overview of the state-of-the-art embedded HDR acquisition solutions. The chapter begins with an overview of existing hardware platforms, which are commonly used for the implementation of HDR acquisition, deghosting and tonemapping tasks, further followed with an overview of specialized hardware accelerators and embedded systems, aimed for high-performance processing. The chapter's last section is devoted to the description of state-of-the-art implementations of HDR acquisition and deghosting on embedded devices and description of theirs, mostly custom based embedded platforms.

3.1 Embedded platforms and accelerators

This section describes selected hardware platforms that are currently available and are related to the algorithms described in the Chapter 2. The HDR processing algorithms can be implemented and even hardware-accelerated on these platforms. Individual platforms are presented and their benefits and disadvantages for such tasks are summarized. The section begins with a definition of metrics used to enumerate the computing power of individual platforms. Subsequently, the platforms based on CPU, GPU and DSP processors are presented, followed by a description of FPGA circuits. To date, very popular are SoC platforms (System on Chip), where the CPU is located on one chip with the accelerator in the form of GPU (nVidia Tegra), FPGA (Xilinx Zynq) or DSP unit (Google PVC).

Performance metrics

Typical performance metrics of computing systems include:

- Computing power
- Memory capacity
- Memory access time
- Power consumption
- Peripheral device support
- Cost

Very important parameters also include development time and overall cost. They are related to the selected platform and its available development, debugging or simulation

tools, the possibility of running an operating system and finally, availability of function libraries.

Large memory and computing requirements are typical for image processing and computer vision tasks. The algorithms are often performing several operations over each pixel. This could be an issue for real-time video processing.

Mainly used metrics are MIPS and MFLOPS, both representing millions of executed instruction per seconds, MIPS is for fixed-point and MFLOPS for floating-point arithmetic instructions.

Algorithms of image processing are often using only simple mathematical operations, such as addition or multiplication. They often do not need a high precision number representation. An 8-bit fixed point or 32-bit floating-point numbers are frequently used for pixel representation. Image processing algorithms are quite well parallelizable in general because the same operations are applied for each pixel/block of the image. This feature allows easy and efficient utilization of parallel computing platforms. It leads to easy computing performance scaling and also to the reduction of cost and power consumption. Then one of the performance metrics could be a number of processor cores or number of parallel operations executed per clock cycle.

Amount of available memory space is often not critical. In most of the cases, one version of the currently processing image have to be held in memory. Some algorithms are requiring more memory space for partition structures and tables storing. Memory capacity metric is a number of bits/bytes, that can be stored in memory. Some image processing algorithms can take advantage of block memory accesses, but on the other hand, some of them need random memory access (mostly algorithms using some partition structures). They have different demands on memory throughput and latency too. Memory throughput is measured on Gb/s. Memory latency shows the time between the start of a memory access request and its finishing. Latency is affected by memory cells speed and mostly by memory system hierarchy. Average memory latency is indicated in nanoseconds (ns).

From the perspective of real system deployment, the overall device cost is one of the most important parameters. This is given by component price, production complexity, development cost and appropriate licence fees. Price is normally expressed in dollars ($). Price is often related to other parameters, such as computing performance (MIPS/$, MFLOPS/$) or memory capacity (Gb/$) due to different platforms comparison.

To compare with different platforms often price applies to other parameters such as computational performance (MFLOPS per dollar) or memory capacity (GB per dollar).

Development price is affected by the available development, debugging and simulation tools and by existing function libraries. It also depends on targeted technology and on knowledge and experiences among developers.

Power consumption becomes an important metric today, specifying how long can device run on battery, the cost of annual traffic etc. Power consumption is primarily affected by integrated circuits parameters, such as manufacturing technology, the size on the chip, clock frequency and supply voltage. Many platforms have power-saving technology like a dynamic clock frequency/voltage scaling and clock-gating. Power consumption is measured in watts (W). Relative units are also used, such as computing power per watt (MIPS/W).

Another important parameter is the support of peripheral interfaces. Each image processing device requires at minimum an input interface, which is used for retrieving image for processing, and output interface for the representation of the results. Additional interfaces are needed, for example, for communication with other devices. Data throughput is the critical parameter for the input interface, especially in real-time systems. Currently, there

are several commonly used and well-supported interfaces as a PCI Express, Thunderbolt, USB, Firewire and custom high-speed serial link/busses.

A separate issue is a possibility of adapting the computing platform for a specific application. One of the possible options is to use external processing units (e.g. for video decompressing). Another option is to use a self-designed computer system. A typical example is application-specific integrated circuits (ASIC) or programmable gate arrays (PLD, FPGA). This technology allows creating an application-specific computing system to the specific application. Such systems have the best parameters relative to consumption, performance and chip dimensions. Their main disadvantage is the very high development cost. System metrics vary depending on the technology used. Unit of equivalent gates was applied for comparison purposes, indicating the amount of AND and OR gates that we can replace with a specific chip.

CPU based platforms

CPUs platforms are designed for general computing. The universal processor allows use in a wide range of applications. CPUs are used as computational units in PCs, servers or some embedded systems. Its main advantage is versatility but counterbalanced with resource requirements (time, power consumption, chip area) comparing to the specialized computing systems. The processor is a complex sequential engine executing algorithms expressed by machine code. The basic part of the processor is the arithmetic logic unit (ALU), cache memory, control unit and IO controller.

Architecture specifics ALU performs operations on the data such as addition, multiplication, division and some logical functions, logical AND, OR and many others. ALU is also driving the program execution, performing a conditional or branching code. ALU is designed to compute with fixed-point numbers only. Floating-point unit (FPU) have to be used for decimal number processing; it is located outside the ALU. Today, in order to reach maximum performance, processors have more ALU units and the FPU is designed for parallel data processing. Processor frequency is today clocked over 4GHz and they can reach the 300,000 MIPS with eight cores and 95W TDP (AMD Ryzen7 1800X). Special server processors have up to 32 cores.

The processor has a fast registry set, where the intermediate results are stored. They operate on ALU frequency and their quantity is limited. On the other side, main memory has huge capacity today, up to 512GB, but it operates on much lower frequencies and with thousand-times bigger latency. This problem is solved by a complex memory hierarchy, where cache memory is embedded to the processor in order to preload and store frequently used memory locations and thus improve memory access latency. The cache is divided to several levels, from L1, which is fastest (latency about 1-3 processor clock cycles) but has the smallest capacity, around 768kB, for L3 with capacity around 8MB (32MB in high-end processors), but with significantly slower access.

Processor's controller unit manages the interaction of individual parts of a processor. Nowadays, it's a very complex and circuit which includes sophisticated algorithms to accelerate processors computing performance by out of order instruction executions, branching code prediction and more.

Currently, there exist two main processor categories applicable to image processing. PC processor architecture is represented mostly by x86 standard compatible processors. Increasingly popular are the ARM-based processors located mostly in embedded systems,

such as mobile phones, tablets etc. The other architectures have a low computing power (embedded systems), minority presence in the market or are not advisable for image processing (server processors).

x86 architecture X86 processors are used primarily in personal computers, notebooks, special embedded systems and servers. They are called CISC processors since they have a lot of very complex instructions in its instruction set. They are based on Intel x86 instruction set, which was first used in the Intel 386 processor and they are fully backwards compatible with this model. The instruction set originally supports only 32-bit operations, but today's processors take advantage of extended AMD64 instruction set with the support of 64-bit operations.

Advantage of these processors is their backward compatibility as the basic instruction set is standardized. This brings an advantage for precompiled software, which is able to run on the newest hardware. On the other hand, backward compatibility is a great bottleneck not only for maximum performance but even to other parameters, like power consumption and even chip size. Overall, they are not quite effective. That's the main reason why there aren't suitable for small and efficiency embedded systems.

ARM architecture ARM processors (Figure 3.1) becomes popular in small embedded systems. Unlike the x86 compatible processors, their instruction set is RISC and without backward compatibility. RISC means that only basic instructions are supported, which leads to small processor complexity. The ARM instruction set is variable, so only certain instructions could be supported. This leads to a small, effective processor with much lower power requirements compared to x86 processors. That's why their popularity is rising, especially in the embedded and mobile segment. ARM processors are sold by ARM holding as a design file, not as a physical chip. It's quite similar to IP cores for FPGAs(described below). Hence the manufacturer can build his own chip with custom peripherals based on the ARM processor core.

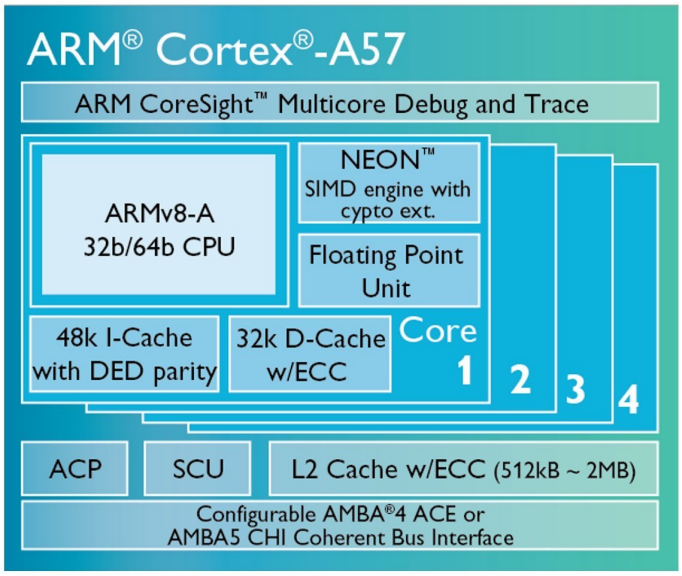


Figure 3.1: Scheme of Quad-core ARM processor A57. Image retrieved from *https://www.androidauthority.com/*

Development software The CPU processors are long supported and widely used so that there exist a huge number of effective development tools in the form of operating systems and their APIs, compilers and libraries, development studios etc. CPU programming is then quite simple and fast, but this applies to one, not to multiple CPU programming. There can be a lot of work with CPU's synchronization. The Windows operating system is primarily designed for x86 processors. On the other hand, Unix based systems are adapted to run on both x86 and ARM architecture.

Applications CPUs are quite suitable for most sequential image processing algorithms. It can effectively handle random memory access, needed by tree or list search algorithms, as well as block memory access. Although the CPU is quite effective in this way, overall computing power should be insufficient. Nowadays, the CPUs are equipped by more than sixteen cores in order to reach high computing performance.

General purpose GPU

GPGPU, i.e. General-purpose Computing on Graphics Processing Unit is the way of using graphic cards GPU's for general-purpose computing instead of their original purpose, performing graphics operations. GPGPU is supported since 2006 by GPU manufacturers.

GPU's have a big computing potential. It is composed of several SMP (Streaming Multiprocessor), each containing a large number of CUDA cores, currently up to 64 per SMP, each running up to 1,5GHz (architecture Nvidia Pascal). Each core can provide basic FX arithmetics and memory access operations. Cores within SMP have access to shared memory, registry set and FP(Floating Point) computing units. Computing cores are simplest as possible, so they don't have their own controlling logic. Cores within SMP are divided into several warps, each with own program controller. Whole warp is then executing one source code, so SMP behaves like the multithreaded processor.

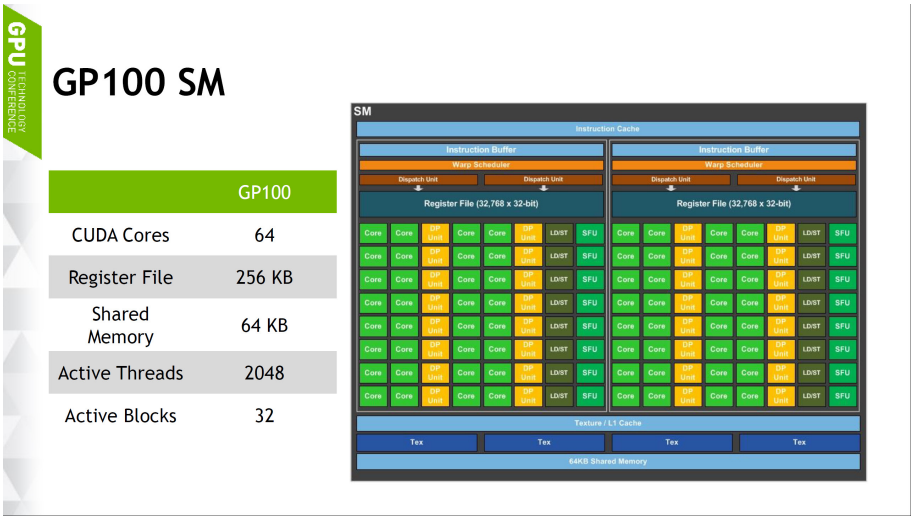


Figure 3.2: Schematics of SMP from nVidia Pascal GPU. Image retrieved from *https://wccftech.com/nvidia-pascal-specs/*.

The graphic card usually contains a large amount of memory on board, currently up to 11GB. To provide as most data bandwidth as possible, multiple cache levels and techniques are applied. All SMP processors share global memory and up to 4096kB L2 data

cache(Pascal). Every SMP have its L1 cache (up to 64kB) shared between all Cuda cores. They also contain a small texture cache that can be used for some special operations.

Architecture specifics In general, GPU has extreme computing power but extreme power consumption too. They can reach up to 10,6 TFLOP, currently with 300W TDP, but could be a serious problem to reach such a performance in a real application. Cores within SMP are further divided to warps which are driven by the common controller, so every core in warp is typically executing the same instruction. If a program contains some branching instructions, as „if-else“ or „case“ statements, the controller has to selectively enable or stop some threads to allow executing of the relevant part of code statement for each core. GPU is capable of running a lot of threads in parallel, so it can process a huge amount of data. The memory subsystem is designed for a high load, but the programmer has to take care of a proper global memory access attitude. Ideally, threads have to gather memory accesses, memory cells required by individual threads should be adjacent. Access with inappropriate stride can rapidly decrease memory throughput. For better performance, access to global memory should be aligned to 128B memory segment.

Shared memory is divided into multiple banks to increase throughput. The proper attitude is to access memory cells by stride 1 or by the stride of the prime number. This results in 1:1 assignment within banks and threads of the warp. Different stride leads to worse ratio and in the worst case to serialize memory requests. In case of reading one memory cell by all threads, exists a built-in memory broadcast mode.

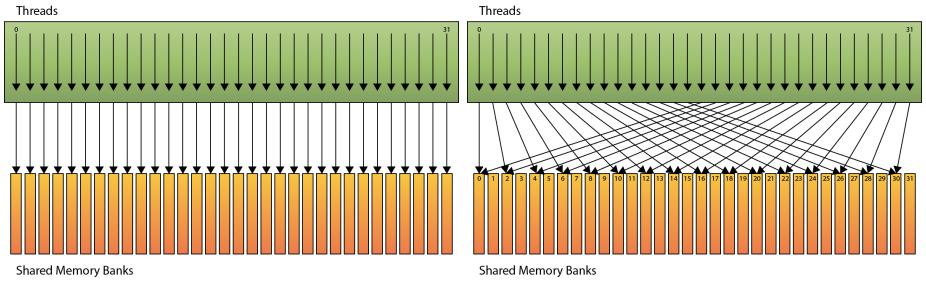


Figure 3.3: Shared memory bank access. On the left is the example of Linear addressing, causing no bank conflict. Two-way conflict is on the right, leads to two memory accesses instead of one in linear addressing.

Development software Currently, there are two programming platforms for GPGPU, Nvidia CUDA and OpenCL. CUDA is developed and supported by Nvidia company, one of the graphic card manufacturers, therefore is available only for Nvidia Graphic cards. CUDA SDK contains large libraries for easier application development. OpenCL is a more general platform and supports GPU of any manufacturer. Moreover, OpenCL can utilize not only GPUs but even CPU for computation. Both platforms are based on C language with additional extensions.

Applications Types of algorithms suitable for running on GPUs are clear from its specific architecture. High performance can be reached on per pixel or local image operations, such as image filters. On the other hand, it can be a problem to write effectively a tree search algorithms, algorithms with random memory access, algorithms with if-else statements etc.

DSP processors

DSP is a shortcut for Digital Signal Processor. It is primarily designed for real-time signal processing. Currently, DSP processors are used for example, at mobile phones (GSM signal coding/decoding), DVD drives, cameras etc.

DSP is based on Harvard architecture and thus has a separated instruction and data memory. This leads to better throughput because every memory has its own bus. To maximize computing power, DSP has several parallel processing units, so in every clock cycle can be executed more than one instruction. Typically, there are more ALUs, MAC Multipliers and data load/store units. DSP also has two or more DAG (Data Address Generator) and a lot of DMA units. DAG unit is capable of advanced memory addressing, such as a round buffer addressing, reverse addressing, stride memory access etc. There are two basic types of DSP, divided by used arithmetics. DSP's are either calculating with fixed or a mix of fixed/floating-point arithmetics. Fixed points performance of one DSP core is currently up to 8 MACS(multiple and accumulate) per clock cycle. Floating-point DSPs have approximately half of this performance, up to 4 FLOPS/clock cycle. L2 cache memory is currently about 4MB per core. TDP of one core is highly dependent on desired performance, from less than 0.1W up to 7W. Nowadays DSP processors can have up to 4 cores and run above 1.2GHz frequency.

Architecture specifics Real-time signal processing is the main utilization of DSP processors, so their architecture is adapted to meet such criteria. One of the great advantages is the presence of multiple computation units capable of parallel processing, which allows easy hardware loop unrolling. DSP is using a long instruction word (VLIW). One instruction is composed of several instructions, each for one processing unit. If a source code contains data-independent instructions, they can be executed simultaneously. This is solved by the compiler, which has to take into account the number of processing units, length of instruction execution and possible data dependencies. This implies that a compiled code is not compatible with different DSP processors. Not all of the processing units can be busy at the time, so the DSP long instruction can contain some NOP instructions. This can have a bad impact on DSP's performance.

The bottleneck of DSP processors is conditional code executing, for example, if-else or case statement. It is solved by preliminary selection of branch code and incidental execution of recovery code. In this case, code cannot be effectively parallelized.

Applications Primary DSP orientation is on real-time signal processing, it's design is quite customized for this purpose. Applications based on FFT, signal coding/decoding, signal compression tasks are very effective on DSP. Algorithms effective on DSP are commonly based on block or stream processing, loops etc. There can be easily achieved parallel execution, loop unrolling and DAG unit utilization. On the other hand, significantly ineffective could be algorithms based on random memory access and if-else and case statements, for example, tree searching algorithms.

Development tools Typically, DSP BIOS is running on DSP chips. It is a kind of real-time operating system (RTOS), which cares about low-level system events, task priorities and provides a simple API for a custom application. C or C++ compilers are available for DSP programming.

3.2 System-on-chip platforms

SoC architectures, sometimes also known as an application services platform (ASP), are a combination of existing architectures described above. Typical hybrid systems are composed of several interconnected chips. Nowaday the CPU + GPU, CPU + FPGA and DSP + FPGA cooperation for signal processing is quite common. A Brand new way is to embed more computing architectures into one chip. Xilinx Zynq SoC and Nvidia Tegra are a typical example of a useful combination of the multi-core ARM processor and FPGA, respectively GPU.

SoC FPGA - Xilinx Zynq

Architecture specifics Xilinx Zynq is a combination of dual-core ARM processor core (and other peripherals, such as memory controllers etc.) with a standard FPGA chip. Both parts can run on different frequencies, so ARM can run on 677MHz and is not decelerated by the slow FPGA clock frequency. Both parts are interconnected through standard AXI bus and share RAM memory. Moreover, the FPGA can be directly connected even to the ARM cache memory. The ARM processor has a 64kB L1 and 512kB L2 cache memory shared between cores. The standard way to communicate between ARM and FPGA is to make a memory-mapped device connected directly to the AXI bus. This device is then configured from the ARM processor and serves as a coprocessor or independent system with access to system memory, like a graphic card on PC. Zynq is an excellent combination of two totally different architectures. ARM CPU is quite suitable for any sequential algorithm. On the other hand, FPGA has a great parallel computing potential but executing of standard sequential code is almost impossible and have to be solved by IP cores processors.

Development tools A big advantage of the Zynq platform is the capability of running a Linux operating system. So there exists a lot of development studios, standalone code compilers and development libraries just like for Linux desktop operating system. Xilinx Company provides support for FPGA part, including IP cores for easy AXI bus connection. They also developed a basic Linux distribution with several examples of interconnecting both parts of a chip.

Applications Several ways how to utilize this platform for some image processing exists. ARM processor can run a complex algorithm using an FPGA part just only to accelerate some computing or graphic operations. This mode is suitable for algorithms, that needs some global information over image or needs random access to the image memory. Another way is to perform the whole algorithm inside the FPGA part. ARM processor can only serve as a driver of some peripheral interfaces, such as Ethernet or USB, whose implementation would be difficult to program in FPGA. An application constraint results from the FPGA subsection above.

SoC GPU - Nvidia Tegra

Nvidia Tegra integrates an ARM CPU, graphics processing unit (GPU), northbridge, southbridge, and memory controller onto one package. The most recent Tegra SoC, Xavier, contains eight custom ARMv8 cores, a Volta GPU with 512 CUDA cores, an open-sourced TPU (Tensor Processing Unit) called DLA (Deep Learning Accelerator). The GPU is ca-

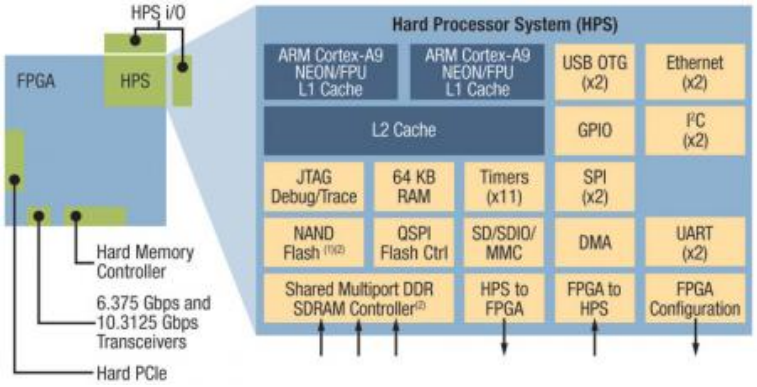


Figure 3.4: A generic example of an SoC FPGA, sometimes also known as an application services platform (ASP), shows a dual-core hard processor system with its complement of hard peripherals on the same die with an FPGA fabric. Image retrieved from *http://archive.rtcmagazine.com/.*

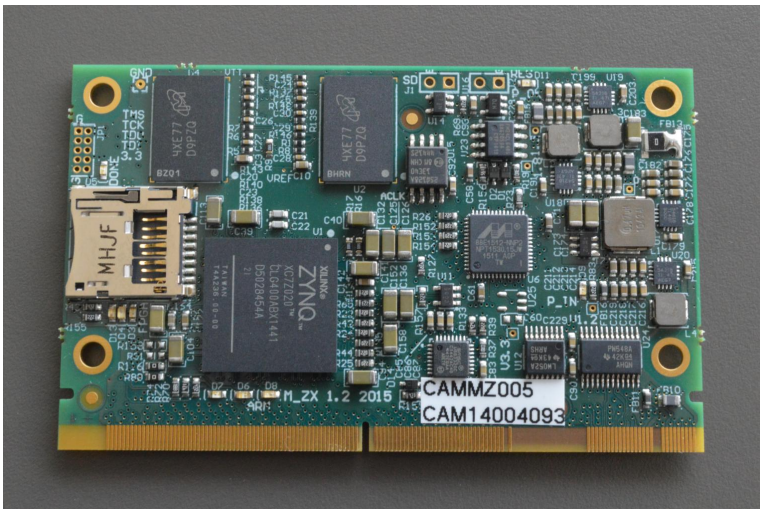


Figure 3.5: Module M_ZX equipped with SoC Xilinx Zynq Z-7020.

pable of encoding and decoding 8K Ultra HD video (7680 × 4320). Users can configure operating modes at 10W, 15W, and 30W TDP as needed.

Development tools NVIDIA devices are supported by the Jetson NVIDIA software stack, enabling to develop once and deploy everywhere. JetPack SDK includes the latest Linux Driver Package (L4T) with Linux operating system and CUDA-X accelerated libraries and APIs for AI Edge application development. It also includes samples, documentation, and developer tools for both host computer and developer kit, and supports higher-level SDKs such as DeepStream for streaming video analytics and Isaac for robotics.

NVIDIA JetPack SDK is the most comprehensive solution for building AI applications. It bundles Jetson platform software including TensorRT, cuDNN, CUDA Toolkit, VisionWorks, GStreamer, and OpenCV, all built on top of L4T with LTS Linux kernel.

NVIDIA L4T provides the Linux kernel, bootloader, NVIDIA drivers, flashing utilities, sample filesystem, and more for the Jetson platform. It is possible to customize L4T

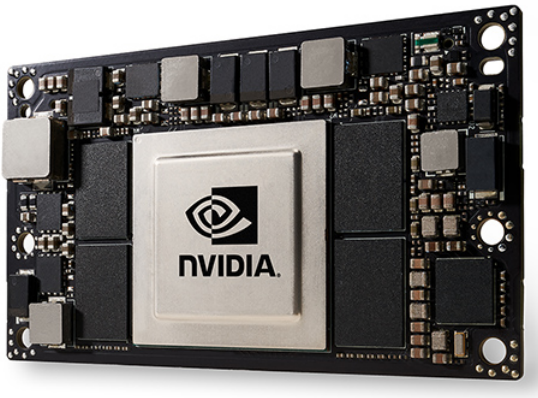


Figure 3.6: Module Nvidia Tegra TX2. Image obtained from *nvidia.com*.

software to fit the needs of the project. By following the platform adaptation and bring-up guide, it is possible to optimize the use of the complete Jetson product feature set.

Applications Tegra GPU supports the applications written in Nvidia Cuda, so many of algorithm written for Nvidia GPUs (e.g. for desktop PCs) are feasible to run on it. Specifically, the Tegra modules are optimized to run deep neural networks. The devices based on the Tegra modules can create powerful edge nodes in the edge-computing scheme. At just 7.5 watts, the Xavier SoC claim to deliver 25× more energy efficiency than a state-of-the-art desktop-class CPU. This makes it ideal for real-time processing in applications where bandwidth and latency can be an issue. These include factory robots, commercial drones, enterprise collaboration devices, intelligent cameras for smart cities.

After consideration, the features of target platforms from Section 3.1, the FPGA was selected as a target platform, namely the SoC Xilinx Zynq, which is a powerful combination of FPGA and dual-core ARM processor on the same chip. This SoC allows the application of hardware-software codesign technique. It brings together the performance benefits of FPGA with the possibility of sequential execution of code - e.g. for driving the FPGA processing or to perform complex calculations, which acceleration in FPGA would be very demanding or not reasonable. Several reasonable arguments for selection of FPGA follows:

- **Power consumption** - FPGAs have very low power in general, and it depends mostly on the amount of programmable resources used.
- **Performance** - Despite its quite low clocking frequency (max. 200MHz), FPGA benefits from parallel processing and achieves very high computational performance. The well parallelizable algorithms, which, e.g. applies the same fixed operations on each pixel/block of the image, are quite often parallelizable to the one result per clock. Thus it is possible to achieve processing speed up to 200MPix/s; of course, the whole pipeline could be extended/multiplied, providing multiplied performance. The increase of complexity usually increases the demands on FPGA resources, not the overall processing speed.
- **ASIC ready** - FPGA implementation is one of the necessary steps during ASIC development. One of the main tasks was an implementation of an HDR compression algorithm (published within the same book[70]), where the „possibility of ASIC

implementation“ is necessary for successful standardization process, required for widespread commercial usage.

- **Low embedding effort** - it is closely related to the ASIC implementability and also the fact that many, predominantly industrial cameras are already based on FPGA, since then the firmware could be easily extended and/or modified. In this case, the HDR acquisition component could fit into existing FPGA located in camera or bigger, quite often with the compatible layout. The implemented FPGA component could also extend the custom ASIC design as a standalone block.

Nowadays, the platforms with powerful embedded GPU, such as Nvidia Tegra, are starting to be concurrent at certain parameters. On the other hand, DSP platforms are slowly getting to the margins of interest.

3.3 State-of-the-art hardware solutions overview

Many research publications were published regarding the acquisition of HDR images; however, only a few of them are oriented on embedded devices. HDR merging itself is not a complex algorithm, but for real-time acquisition, it requires a high memory throughput and external memory buffer, which is not available on many embedded platforms.

FPGA based platforms are more than suitable for such type of applications. Several papers focused on FPGA acceleration and related to our work were published [25, 27, 61, 62, 30, 45, 68, 56]. This section provides its overview and presents achieved properties.

Realtime HDR video for eyetap wearable computer by Mann et al.

Mann et al. [30] developed an FPGA based wearable HDR seeing aid designed for the electric arc welding (see Figure 3.7). The prototype consists of an EyeTap (electric glasses) welding helmet, with a wearable computer upon which are implemented a set of image processing algorithms that implement real-time HDR image processing together with applications such as mediated and augmented reality. The HDR video system runs in real-time and processes 120 frames per second, in groups of three or four frames. The processing method, for implementation on FPGAs (Field Programmable Gate Arrays), achieves real-time performance for creating HDR video using the novel compositing methods, and runs on a miniature self-contained battery-operated head-worn circuit board, without the need for a host computer. The result is an essentially self-contained miniaturize hardware HDR camera system that could be built into smaller eyeglass frames. [30]

Mann's proposed method is adapted specifically for direct hardware implementation, as opposed to assuming the availability of a multi-core CPU or GPU. Additionally, they present how this method can be extended to three or more images in extreme dynamic range cases using simple binary operators. Mann proposed a novel computational method using LUTs (lookup tables) to compute the HDR (high dynamic range) video in real-time. For the case of compositing two images with 8-bit colour depth per channel, a simple size $256 \times 256 \times 3$ LUT can be derived for each camera. The LUT need only be computed once each time a new camera is plugged in for the first time.

The HDR output values are precomputed for a full range of input pixel combinations and stored in lookup tables in BRAMs (see Figure 3.8). Even after certain optimizations of memory consumption, the BRAM demands are very high, especially when more than two

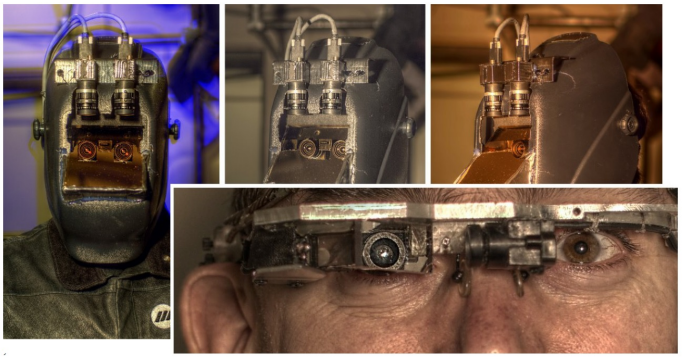


Figure 3.7: The "MannVis welding helmet" implements the EyeTap principle which causes each eye to, in effect, function as if the eye itself were both a camera and display. Image obtained from [30].

LDR images are used. The system is implemented on Spartan-6 LX45 FPGA and produces 720p video at 60 FPS while fusing two images.

Mann's prototype have the camera configured to capture images that are four stops part (i.e., one image has an exposure time that is $2^4 = 16$ times longer or shorter than the other). Once they estimated from the image set, we perform dynamic range compression (tone mapping) for LDR display.

For the case of constructing HDR images from 3 or more images, they can compute an intermediate estimation of the photographic quantities, since the images only differ in exposures (i.e., four stops in proposed solution), the same LUT which precomputed quantities can be applied to the image pairs at no additional computational cost. [30]

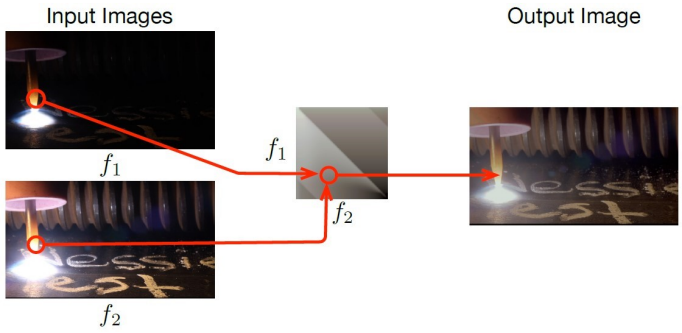


Figure 3.8: The figure illustrate the direct lookup method used by Mann et al [30]. The pixel values from the same pixel in the image addresses the precomputed resulting HDR values in the table (in the middle). Image obtained from [30].

In order for real-time HDR processing to be practical, a 45nm low-power Spartan-6 LX45 FPGA device was selected for its low power consumption and portability. The board contains two input HDMI ports used to receiving the baseband HD video (720p@60 FPS) and two output HDMI ports used for transmitting the processed HDR video frames. It also contains 128MB of DDR2 SDRAM, used for storing video frames. It runs at 625MHz in order to meet the real-time processing requirements. Additionally, BlockRam (BRAM) is used as line buffers and to store the LUT. However, it is limited to a capacity of 2.1Mbits

(116 x 18,432). Due to its resource limitations, a focus on reducing complexity guided Mann et al. [30] to invent novel approaches to HDR video. Much of the processing is precalculated and stored in a lookup table to reduce complexity using the methods discussed in the Mann's Article [30] each pixel sample contains an 8-bit RGB colour component totalling 24-bits per pixel. Each lookup is addressed by a colour channel of two differently exposed frames. A total of 16-bits is used for addressing into the LUT totalling 65536-entries. For 8-bit wide sample output, BRAM can be configured as 8-bits wide × 2048-deep, which resulted in 32 BRAMs utilized per colour channel for a total of 96 (out of 116) BRAMs utilized. Efficient on-chip memory utilization is the key to LUT implementation technique, as it directly limits scalability beyond two frames. The number of BRAMs required can be reduced by utilizing the fact that only half of the data in one square LUT is valid since each pair of frames has one's pixels always greater than the others. This technique is used in 3-frame implementation. [30]

Realtime HDR Video Imaging on FPGA by Tao et al.

Tao et al. [57] extended the work of Mann [30] by introducing a lookup table compressed using quadtree structure, which saves the amount of BlockRAM resources. Tao replaced the weighted sum approach with the new quadtree-based compositing for high-quality HDR video production. The proposed compositing circuits are generated by the software, with parameters given by the user. It compresses and implements a 2D Lookup Table (LUT) on an FPGA, by bounding the error and space of quadtree representation of the original LUT according to the expected usage, so that the LUT is compressed to fit within the total amount of the block RAM resource available in a mid-sized FPGA. They also add the support for 1080p video at 60 FPS. [57]

HDR-like imaging using industrial digital cameras by Popadic et al.

Popadic et al. [45] proposed a low complexity method for capturing high dynamic range scenes using standard industrial digital cameras. The goal of the proposed method is to improve the performance of the standard industrial digital cameras, without modifying the central processor unit (CPU) software. The most effective way from their point is to avoid depending on the camera software manufacturer (that do not allow changes in the software) is to use the FPGA to perform operations that extend the dynamic range of the images and output them in a format compatible the CPU. The FPGA unit is designed as independent and intended to be connected transparently between CPU and CMOS chip of camera. To keep the compatibility, the FPGA unit produces an HDR-like image instead of HDR. As opposed to HDR image format, which uses 32-bit floating-point precision for each colour channel, HDR-like image is stored in standard 8-bit RGB format. Overall, the performance of the system is much enhanced, while more details in the resulting image are shown. Moreover, the standard HDR image cannot be shown on standard displays or printed using standard printers; HDR has to be post-processed by tone-mapping operators. Such produced HDR-like image has not these shortcomings and may be processed in the existing image processing pipelines. [45]

In the proposed architecture, sensor and FPGA together form a „smart sensor“, which receives configuration data from the CPU. In order to provide an HDR-like image, three images are combined into one. In order to combine them in a suitable way, one has to determine weight coefficients. Communication between CPU and FPGA is realized by the same interface as the sensor does. CPU initially calculates auto-exposure time as part of its

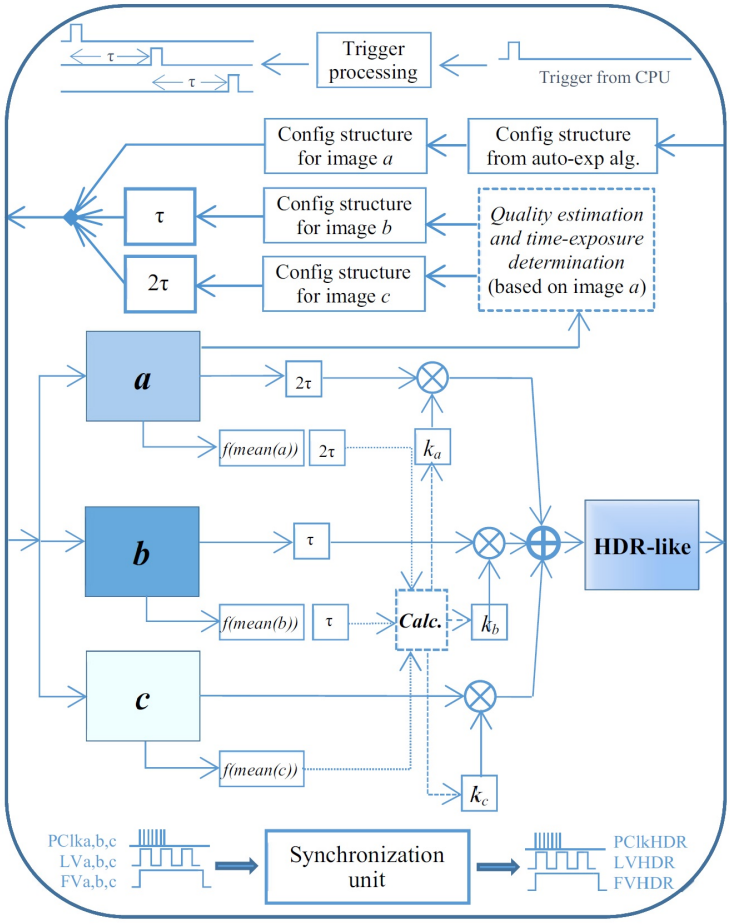


Figure 3.9: Method for HDR-like imaging proposed by Popadic et al. [45], implemented on FPGA. Image obtained from [45].

auto-exposure algorithm. Structure of the method implemented is presented in Figure 3.9. The method is implemented on the FPGA embedded onboard, which represents a bridge between CPU and sensor. Image acquisition is triggered after setting the configuration parameters that are prepared for a certain image. HDR-like image, the output from the FPGA will be automatically accepted by CPU. It is represented in the same format (Bayer matrix format) as a RAW image from the image sensor. Image fusion pipeline implemented in the FPGA will be presented later. Since standard digital cameras are equipped with an auto-exposure function, the first step of the proposed method is to estimate the quality of a single image. When the camera process captures the auto-exposed image, image quality should be checked. If a single image quality is not acceptable according to the proposed criterion, auto-exposure time is not optimal. In order to improve image quality, the process continues to the HDR-like image generation by taking another two images. The second step of the proposed method is an algorithm which calculates exposure times of two additional images that will participate in the final image. The third step of the method is an algorithm which performs a fusion of three obtained images. Proposed algorithm for HDR-like scene-mapped imaging performs calculations on the global image level. Smoothing is not necessary. This approach makes the algorithm simple and fast. [45]

HD WDR video surveillance system based on FPGA by Xie and Wang

Xie and Wang [68] developed a high definition wide dynamic video (WDR) surveillance system, based on HDR camera kit accompanied by Lattice FPGA and sensor Panasonic MN34229. The WDR surveillance video is displayed through the HDMI port. They use the algorithm of exposure fusion, which introduced Mertens et al. [34] and fuses two images into a wide range image only. The whole WDR module can be divided into four parts: luminance value statistics, weight calculation, mean filter and pixel fusion, as shown in Figure 3.10.

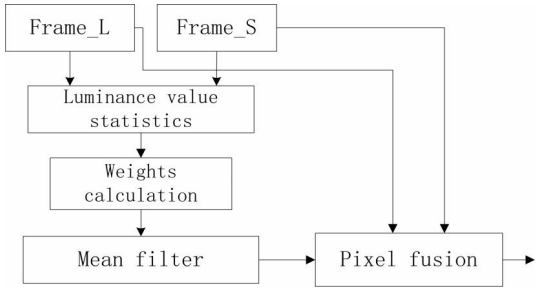


Figure 3.10: System block diagram of the method for WDR imaging proposed by Xie et al. [68]. Image obtained from [68].

The video image data is collected line by line. To ensure the real-time performance of the system, they calculate the fusion weight of the current frame for the multiple exposure image synthesis of the next frame. First, when the data of the nth frame arrives, it is divided into a number of image blocks in 32×32 size, and the brightness values are count from each image block. The sum of all the pixels' luminance values in each image block of long and short exposure image is stored in RAM. In this way, They can get each image block's sum of the luminance values in the nth frame at the end of the image. Further, they can calculate the fusion weights through the existing mathematical model, and the results are stored in the shift registers in turn to complete the mean filter operation. The function of a mean filter is to reduce the block effect. After the mean filter is completed, the final fusion weights are stored in RAM waiting for the pixel fusion. It's worth mentioning that weights calculation and mean filter is completed during the time between the nth frame to the next frame. When the data of the next frame arrives, the fusion weights we just got are used for pixel fusion and the WDR data is obtained. The system produces video of 1920×1080 pixels at 30 FPS. [68]

HDR-ARtiSt: an adaptive real-time HDR smart camera by Lapray et al.

Lapray et al. [25, 26, 27] developed a complete FPGA-based smart camera architecture named HDR-ARtiSt (High Dynamic Range Adaptive Real-time Smart camera). This smart camera is able to provide a real-time HDR live video from multiple exposures capturing to display through radiance maps and tone mapping. The main contribution of their work is the generation of a new FPGA embedded architecture producing an uncompressed Black&White 1280×1024-pixel HDR live video at 60 FPS. An embedded DVI controller is also provided to display this HDR live video on a standard LCD monitor. The HDR-ARtiSt camera could obviously embed some complex image processing applications onto the FPGA or could be connected to a more standard PC managing the video stream. [27]

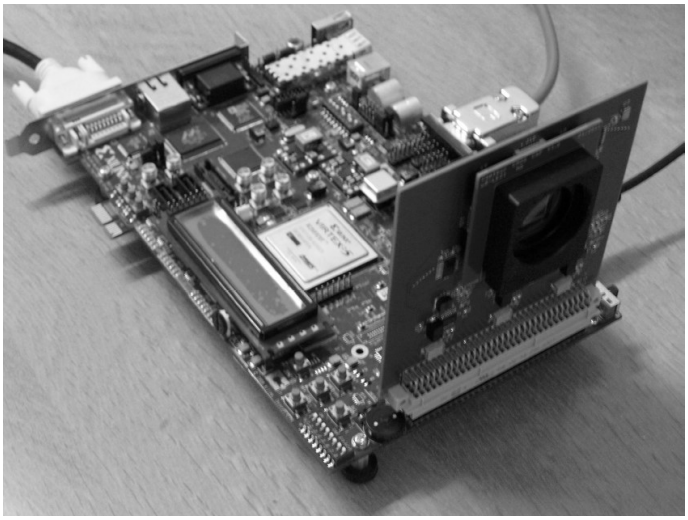


Figure 3.11: Xilinx Virtex-5 ML507 FPGA board equipped with 1.3MPix CMOS, where Lapray et al. [25, 26, 27] implemented the HDR image acquisition and tone mapping. Image retrieved from [27].

To compute real-time HDR algorithms, a dedicated FPGA-based smart camera architecture was designed to address the computation capacity and memory bandwidth requirement (see Figure 3.13). This architecture does not put any restriction on the number of frames used for HDR creating. They shortly call each architecture derived from the generic one as an HDR-P, where P is the number of frames. [27]

According to the detailed description of these methodologies and the comparison of their real-time software implementations, they decided to use the Debevec's method [5] for HDR merging. The main advantage of this approach is that there is very little constraint about the response function (other than its invertibility). Moreover, the proposed algorithm proved to be quite robust and easy to use due to the simplicity of Debevec's equation (see Equation 2.3). The HDR creating pipeline for HDR-2 video is shown on Figure 3.12. [27]

Regarding the tonemapping operators, Lapray et al. [25, 26, 27] implemented two the global tonemapping operators by Duan [8] and Reinhard [47]. Their implementations were published and described thorough their articles [25, 26, 27].

The HDR-ARtiSt platform [27] is a smart camera built around a Xilinx ML507 board, equipped with a Xilinx Virtex-5 XC5VFX70T FPGA (see Figure 3.11). The motherboard includes a 256 MB DDR2 SDRAM memory used to buffer the multiple frames captured by the sensor. Several industry-standard peripheral interfaces are also provided to connect the system to the external world. Among these interfaces, the vision system implements a DVI controller to display the HDR video on an LCD monitor. It also implements an Ethernet controller to store frames on a host computer. [27]

A custom-made PCB extension board has been designed and plugged into the FPGA board to support the Ev76c560 image sensor, a 1280×1024 pixel CMOS sensor from e2v company. It offers a 10-bit digital read-out speed at 60 FPS in full resolution. It also embeds some basic image processing functions such as image histograms, evaluation of the number of low and high saturated pixels. Each frame can be delivered with results of these functions encoded in the video data stream header. [27]

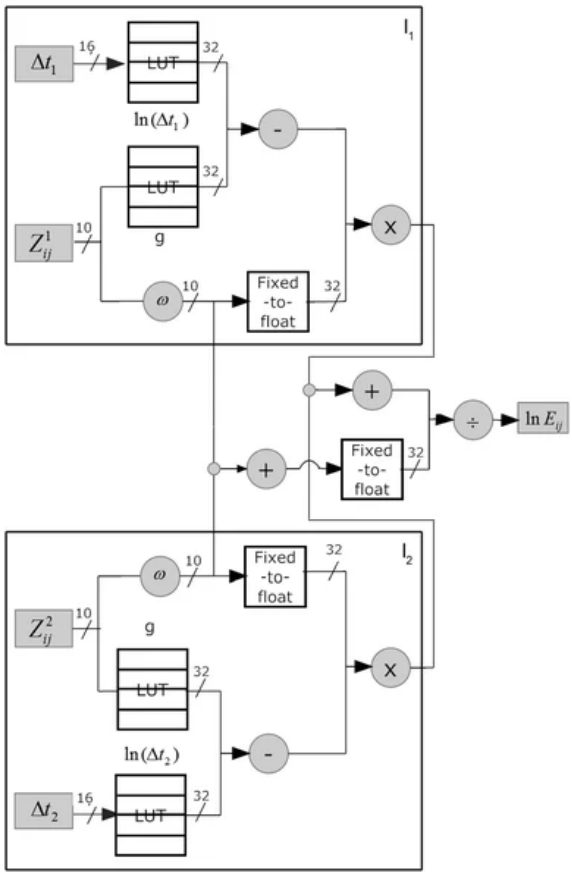


Figure 3.12: HDR creating pipeline using LUTs tree for merging two input images, implementing the Debevec's algorithm [5]. Image retrieved from [27].

Multi-streaming memory management unit the MMU-P can capture and store the current stream of pixels from the sensor and delivers simultaneous $P-1$ pixel streams previously stored to the HDR creating process. With such memory management, they avoid waiting for the capturing of new P frames before computing any new HDR data. Once the initialization is done, the system is synchronized with the sensor frame rate (i.e., 60 fps) and can produce a new HDR frame for each new capture. Moreover, in terms of memory, the MMU-P requires to store only $P-1$ frames, because the oldest captured frame is read and overwritten by the current frame acquired by the sensor. For reasons of efficiency, the MMU-P reads and stores lines of pixels.

3.3.1 Real-time HDR video compression using an FPGA by Zemcik et al.

The architecture of the HDR camera proposed by Zemcik et al. [70] can capture 30 FPS FullHD with each frame formed from two exposures, or 20 fps FullHD video formed from three exposures. With sharing the expositions, the output can eventually reach up to 60FPS; however, the whole pipeline is limited by the capability of H.264 encoders, supporting 30FPS only. The main architecture highlight is the encoding of HDR video using two standard video codecs.

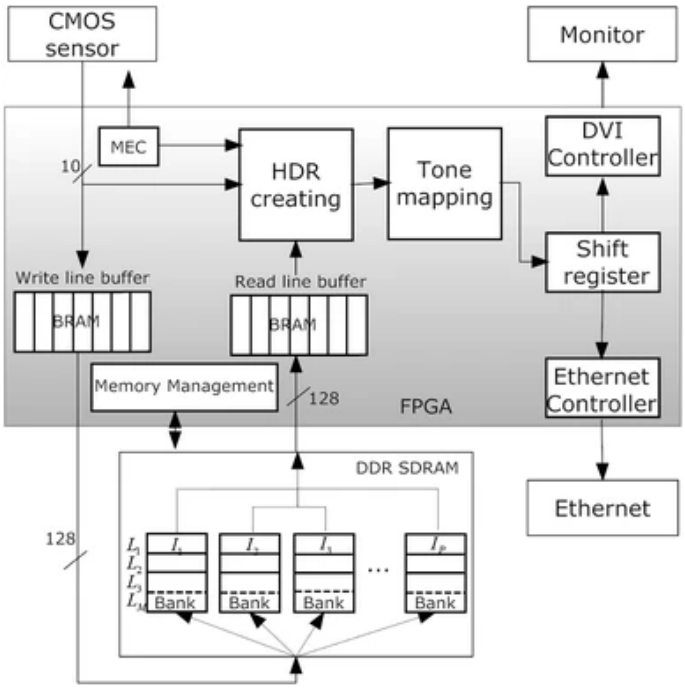


Figure 3.13: overview of the HDR-P video architecture, designed by Lapray et al. [27]. Image retrieved from [27].

This architecture uses standalone 2K Flare[1] camera (see Figure 3.14) connected over 3G-SDI interface[2] (commonly used in TV studios). This camera is producing high quality FullHD RAW image at up to 60FPS.

Architecture by Zemcik et al. uses two or three images for HDR merging, depending on configuration, so there is implemented the equivalent number of framebuffers. The double buffering technique is used to avoid rising of image artefacts, which doubles the memory requirements but prevents rising of image artefacts.

All three framebuffers are read synchronously by multiple DMA channels; here, the HDR merging takes place. The read-out is performed once the three images are captured, the output then has 1/3 of the input FPS. Unlike the Lapray [27], which is merging HDR from three last images and thus have the equal input and output framerate, the architectures presented in Zemcik's article [70] took advantage of the speed of the attached camera (capable of 60 FPS) and capture three images as fast as possible, trying to reduce the ghost artefact to the minimum.

[1] http://www.ioindustries.com/
[2] https://www.smpte.org/standards

Figure 3.14: Flare SDI Camera, image retrieved from http://www.ioindustries.com

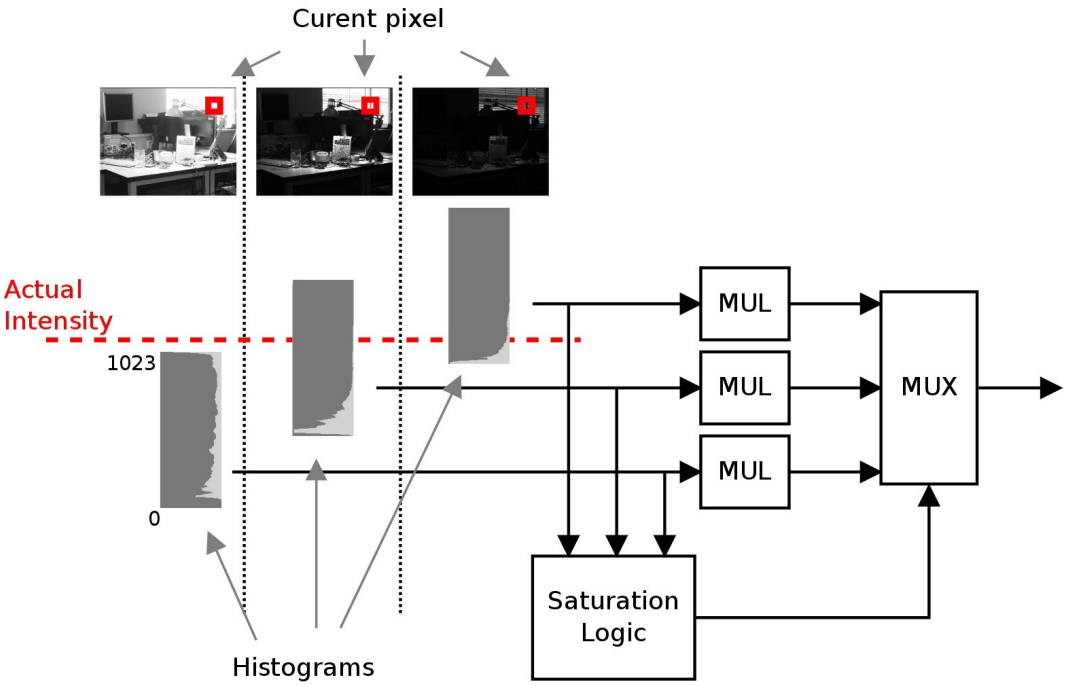


Figure 3.15: Scheme of HDR merging algorithm based on unsaturated pixel selection [70].

The primary demand for the HDR merging algorithm was the capturing of the as-high-as-possible dynamic range, showing the benefits of HDR acquisition. Regarding that the architecture Zemcik et al. [70] use only a simple pixel selection algorithm (see Figure 3.15) because the exposition times are set so far from each other (by multiples of eight), that the particular pixel is exposed well only in one exposition. The others are often under on overexposed; thus their contribution to computed HDR value would be marginal.
The image acquisition is typically made in a colour format where the intensity component is easy to process in HDR chain while the colour is preserved independently. In this case, the YCbCr model is used.

HDR compression The standard dynamic range image and video are easy to compress and reduce their size dramatically, however many standard algorithms (JPEG, MPEG, H.246, ...) are designed only for 8-10bit images. The straightforward solution they proposed is to compress the image or video right after tonemapping, where the HDR image information is compressed to lower bit-width, keeping all important visual image details. Unfortunately, the tonemapping process necessarily means a loss of HDR information.

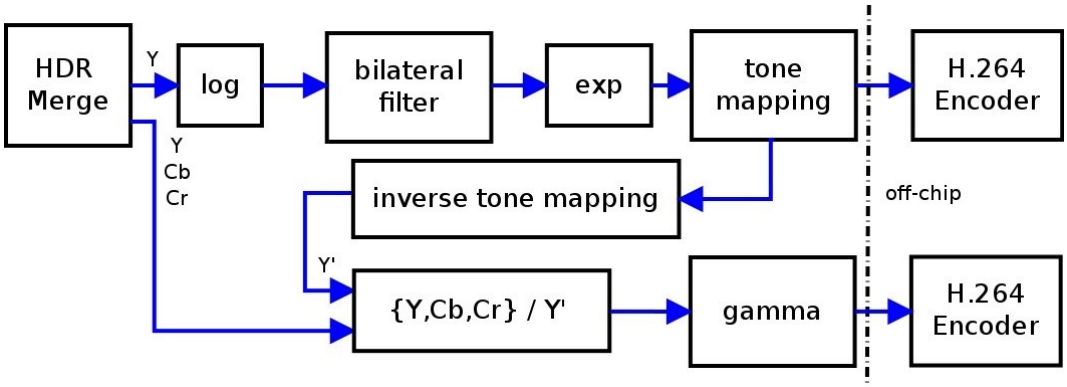


Figure 3.16: HDR image compression scheme [70]. The bilateral filtering in logarithmic domain is performed on luminance channel of HDR image. The output of filtering (*base layer*) is formed into first image stream and the second stream is created as difference between filtered image and original HDR(*detail layer*).

Compression of tonemapped image attempts to keep the important image details; however, compression necessarily means a loss of part of HDR information. Zemcik's proposed architecture for HDR compression [70] uses the bilateral filter (BF) for *base* and *detail* layer separation (which is also first step of Durand operator [9]). The bilateral filter is a very demanding operation with the quadratic dependency of computing operations on filter kernel size. The original algorithm from book [70] requires kernel at least of 19x19 pixel size. This size of BF would be very demanding on FPGA resources, and then the kernel size was selected based on the demands to provide the highest filtering quality as possible while keeping reasonable computational complexity and also reasonable FPGA resource demands. The best choice for the desired implementation seemed to be the BF kernel size of 11x11 pixels. Furthermore, the multiplication of pixels with coefficients was converted into simple addition of shifted operands, saving the resource of DSP blocks of FPGA. Moreover, the experiments proved, that number of additions can be limited, favouring the most significant bits, with marginal impact onto numerical precision, which is summarized in an article by Nosko et al. [42].

HDR camera demonstrator The HDR camera designed by Zemcik et al. [70] is shown in Figure 3.17. The block diagram in Figure 3.18 shows the main building blocks of the camera hardware, which heavily relies on programmable hardware, specifically the Xilinx Zynq architecture that combines the FPGA technology with a couple of ARM CPUs (see Section 3.2). This system core is accompanied with the H.264 video compression chips that accomplish the standard task of 8-bit video compression that is a part of the proposed compression scheme[3]. The complete HDR camera is completed with several electrical in-

[3] http://www.fujitsu.com/us/products/devices/semiconductor/H.264/mb86m01-2-3.html

terfaces, such as SDI, and while it can run from battery power, it is also accompanied by a battery power supply and charging blocks.

The SDI and also many other high-speed video interfaces are based on high-frequency serial buses, which requires high-speed transceivers on the FPGA side. Even though its relatively high price, we chose the Xilinx ZC706 board, equipped by SoC Zynq XC7Z045[4] to build this HDR camera prototype. In the time of development, it was the only reasonable choice within the boards with high-speed GTX transceivers.

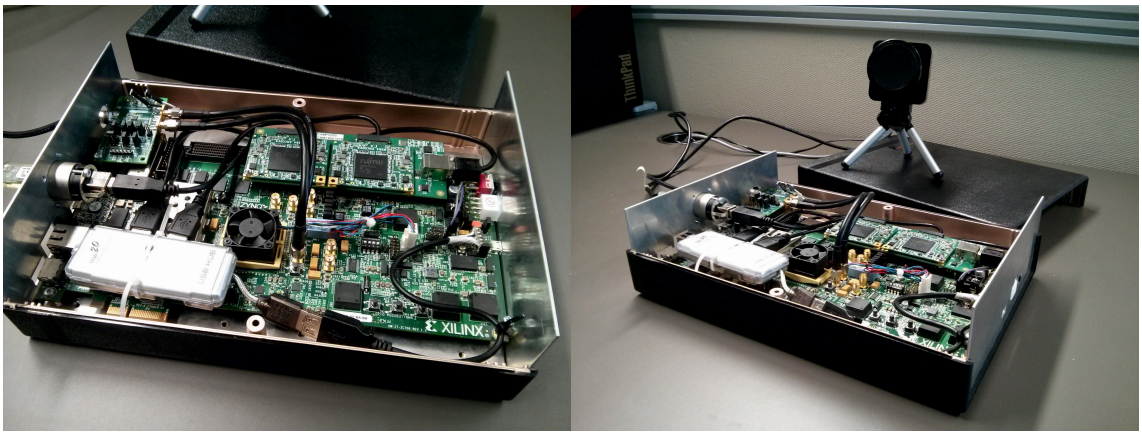


Figure 3.17: A photography of the HDR camera prototype. Note, please, the FPGA development board, the compression modules, and also the Flare camera connected by SDI interface. Image retrieved from Zemcik et al. [70].

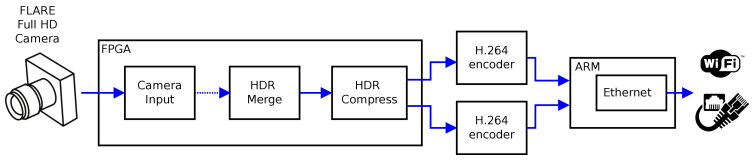


Figure 3.18: Overall scheme of HDR Acquisition architecture published in [70].

3.4 Ghost avoiding/removing solutions

The following section summarises the state-of-the-art HDR acquisition solutions, which either suppress and remove ghosting effect or prevent its occurrence.

HDR camera based on dual-gain CMOS by Tang et al.

Tang et al. [56] developed an HDR camera based on Altera FPGA and equipped with dual-channel CMOS GSENSE400BSI, which is able to apply different analogue gain to the same captured data (see the prototype on Figure [56]). The HDR camera can capture the wide dynamic range image of the nature scene without ghosting phenomenon, by combining the two images with different gain to an HDR frame up to 95 dB. Additionally, the frames are captured at the same moment by two channels with different gain, which reduces the

[4] https://www.xilinx.com/products/boards-and-kits/ek-z7-zc706-g.html

interference between successive frames. However, the CMOS sensor has a rolling shutter, and the disruptive effects can still occur. [56]

The dual-gain CMOS sensor uses two parallel gain amplifiers followed by ADCs to convert the input pixel into digital information (see Figure 3.20). In the HG (High gain) image, it clearly records the details of the dark region of the target scene but loses the details of the bright region. On the contrary, the details of the bright region of the target scene are recorded clearly, but the details of the dark region are lost in the LG (Low gain) image. To obtain more useful information of HG images and LG images at the same time, we can take advantage of the complementary relationship between HG images and LG images, especially when there is a large gap between the target brightness and background brightness in some scenes. [56]

In such way of HDR acquisition avoids rising of ghosting effect caused by sequential image acquisition; however, the CMOS sensor has a rolling shutter, and then another kind of image artefacts still occurs. The maximum frame rate of the camera is 60 FPS at a resolution of 1920×1080. The camera uses a global tone mapping operator by Duan et al. [8].

Figure 3.19: The hardware platform of the camera by Tang et al. [56]. Image retrieved from [56].

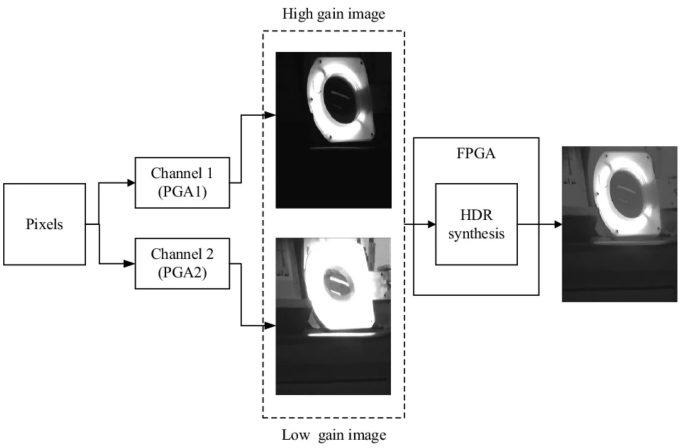


Figure 3.20: The system block diagram of the camera by Tang et al. [56]. PGA – Programmable Gain Amplifier. Image retrieved from [56].

Real-time ghost-free HDR video using weight adaptation by Bouderbane et al.

Bouderbane et al. [4] implemented a deghosting algorithm on the same platform as Lapray et al. [27]. Their method repose on the modulation of weights of the Debevec [5] algorithm, where they adjust pixel weights based on their deviation from pixels of the reference image using the weighting function given [3] (see Figure 3.21) which parameters are taken from Sidibe et al. [51].

To calculate final weighs (Figure 3.21 right) to be used in the high dynamic range reconstruction, they multiply the standard weights from Debevec [5] by the modulation factor (Figure 3.21 left).

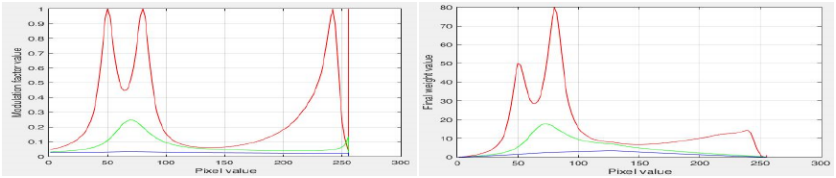


Figure 3.21: The weight modulation factor (left) and the final weight function(right) used in ghost removal HDR merging. Red curve is the factor for the closest radiance value of LDR images to the reference radiance value, the blue curve is the farthest value from the reference value and the green curve is for middle values. Image retrieved from [4].

The implemented FPGA design (including the pipeline of Lapray et al. [27]) uses 29% of the Xilinx Virtex-6 (xc6vlx240t) FPGA and the maximal clocking frequency is 114MHz.

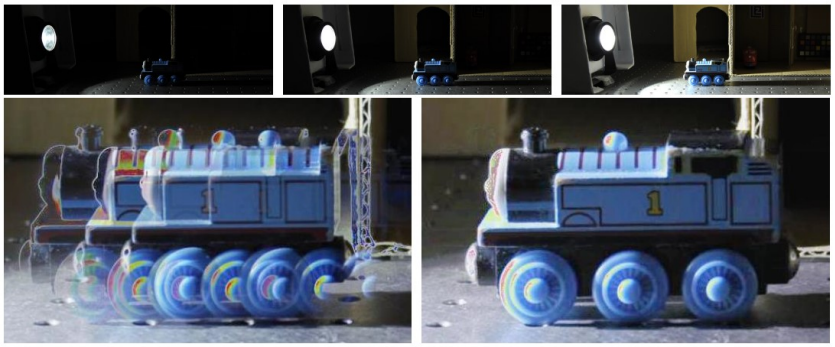


Figure 3.22: Results of deghosting method proposed and implemented by Bouderbane et al. [4]. Image retrieved from [4].

The camera is based on a Xilinx ML605 platform board. With FPGA Xilinx Virtex-6 (xc6vlx240t). The ML605 board integrates a 512 MB SDRAM used to buffer the sequence of three images, which are managed by the memory management unit of Lapray et al. [25, 26, 27]. The attached PCB module integrates an E2V CMOS with a resolution of 1280×1024 pixels. The Ethernet interface is used to stream the HDR images to a host computer with two modes, 32 bits images at 15 FPS (limited by 1Gbit Ethernet) or 8 bits tone mapped images at 60 FPS (limited by a sensor), both modes in full resolution. The DVI output interface is used to display tonemapped HDR frames on a monitor.

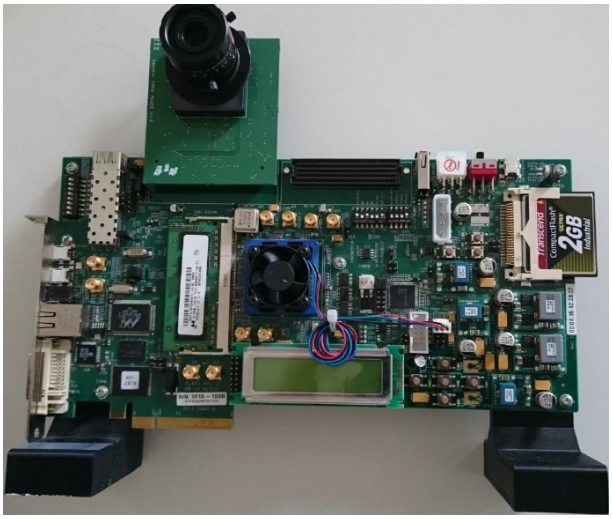


Figure 3.23: Xilinx Virtex-6 FPGA development board equipped with colour CMOS, where Bouderbane et al. [4] implemented the HDR image acquisition and deghosting algorithms. Image retrieved from [4].

True HDR camera with bilateral filter based tone mapping by Nosko et al.

Nosko et al. [41] published the HDR architecture implemented on a custom camera platform based on SoC Xilinx Zynq XC7Z020 (see Figure 3.24). The platform is equipped by a low noise global shutter CMOS sensor Python2000 from ON Semiconductor, connected directly to FPGA through high-speed LVDS (Low Voltage Differential Lines) interface. The CMOS has a resolution of 1920×1280 pixels. The camera provides up to 30 FPS of grayscale HDR video with fixed f-stop range. However, the architecture itself is capable of processing up to 96 FPS. The architecture implements a high quality local tonemapping operator by Durand [9] based on the bilateral filter of 9×9 pixels. Resulting tonemapped image is streamed over the network in the form of MPEG2-TS stream.

The HDR camera architecture published by Nosko et al. [41] is based on the method by Debevec [5]. The exposition weights for individual images are calculated as follows: Given the image with shortest exposition t_1 time weight equal to one, the other images will be given the weights of $\frac{t_i}{t_1}$, where t_i is exposition time of ith image in sequence. The HDR pixel value is computed as follows:

$$H = \frac{\sum_{i=1}^{n} L_i \cdot w(L_i) \cdot \frac{t_i}{t_1}}{\sum_{i=1}^{n} w(L_i)} \tag{3.1}$$

where is the HDR pixel value, L_x is the x-th image in the sequence, t_i exposition time of i-th image and w the weighting function 2.1).

Unlike the algorithm by Debevec [5] they chose a plateau weighing function (see Figure 2.1) as the one leading to the best visual experience; however, it can be easily customized.

The exposition time of the middle image in the sequence is configurable, however, the mutual intervals between exposures are fixed to multiples of two, which leads to shift operations instead of multiplication. Only a middle exposition value is configurable [41].

The resulting HDR pixel is obtained by dividing the sum of pixels by sum of weights. The division is a time and resource-demanding operation, so Nosko et al. [41] decided to

Figure 3.24: Prototype of HDR camera by Nosko et al. [41, 42]

convert it into multiplication by a tabulated fractional value. The sum of weights, according to bit-widths of intermediate results, needs to be represented by 11 bits (sum of three 9 bit values fits into 11 bits), so the fraction value is tabulated on 2048 entries. The resulting HDR pixel is in 10.8 fixed-point representation.

HDR camera prototype

The HDR camera developed by Nosko et al. [41, 42] (Figure 3.24) is based on Xilinx Zynq architecture that combines the FPGA technology with a couple of ARM CPUs (see Section 3.2). The camera is accompanied with the H.264 video compression chip[5] that accomplish the encoding of the video output in the form of HDR tonemapped video.

The sensor is attached directly to the FPGA through high-speed multi-line LVDS interface, which does not require the presence of high-speed serial GTX transmitters on the chip and therefore allows the use of low-cost Zynq XC7Z020. This FPGA has quite limited resources, but the experiments revealed that the resources are sufficient, even for implementation of advanced local HDR tonemapping.

colour HDR video processing architecture for the smart camera by Nosko et al.

This architecture further improves the architecture by Nosko et al. [41]. The architecture provides up to 30 FPS of colour HDR video with fully adjustable f-stops. However, the architecture itself is capable of processing up to 96 FPS. The architecture implements particularly a ghost removal algorithm and a high quality local tonemapping operator by Durand [9] based on the bilateral filter of 11×11 pixels.

The architecture is further enhanced by a colour support. They process individual pixels of colour Filter Array (CFA), in this case, a Bayer mask, in the same manner as the grayscale pixels [55]. The colourization of the HDR image is done later, during the tonemapping process.

HDR merging with ghost-free extension The ghost-free HDR merging is based on the prediction of the pixel value. It is based on similar principles as the solutions of Grosch [15], Wu [66] and Wang [63].

[5] http://www.fujitsu.com/us/products/devices/semiconductor/H.264/mb86m01-2-3.html

Since the exposure time of each image is known, individual pixel values in image i can be predicted using values from j.

$$L_i \approx L_j \cdot \frac{t_i}{t_j} \tag{3.2}$$

where t_x and t_y are exposition times of images. This relation holds only for non-saturated patches, over/under-exposed patches must be handled differently. The ghost detection is performed before the HDR merging phase, resulting in ghost pixel mask (further called as *ghostmap*), where the marked positions are treated differently from the non-marked ones during the HDR merging.

The function Ω tests the two images whether their pixels follows the prediction:

$$\Omega(L_i, L_j) = \begin{cases} 0 & L_i \cdot \frac{t_j}{t_i} / \alpha > L_j \\ 0 & L_i \cdot \frac{t_j}{t_i} \cdot \alpha < L_i \\ 1 & \text{else} \end{cases} \tag{3.3}$$

where α represents the tolerance, which must be taken into account, since the sensor noise, quantization errors and CRF precision may influence the predicted value and thus cause the false ghost detections. According to the experiments, they use the tolerance $\alpha = 1.2$ as default, but this value can be adjusted based on the sensor features. In general, decreasing tolerance leads to more strict ghost detection, where more pixels are marked as ghosts, which eventually leads to worse dynamic range recovery. Increasing the ratio, on the other hand, decreases the chance of successful ghost detection. The *ghostmap* is defined as follows:

$$\overline{G} = \prod_{i=1}^{N-1} \Omega(L_i, L_{i+1}) \tag{3.4}$$

where non-zero value of G marks ghost pixels. The algorithm description is simplified by pixel range control - all of the under/over-exposed pixels are omitted from the value prediction; Still, they are tested for extreme luminance changes (dark to bright and vice versa). The algorithm works per-pixel and uses simple arithmetic operations, and thus it is suitable for implementation on FPGA. The follow-up HDR merging algorithm is modified and in the areas, where *ghostmap* indicates motion, incorporates the pixels from only one image, called reference image, which is generally the best exposed or middle exposed image in the sequence.

Nosko et al. [42] implemented the HDR merging algorithm from Debevec [5] with modification for Ghost removal. Every pixel is assigned with the weight w, calculated using the triangle weight function by Debevec [5]. The function was modified for the shortest and the longest exposure since in the original algorithm, the saturated pixels are assigned a value darker than pixels not completely saturated.

The HDR image H is a weighted sum of corresponding pixels in sequence of L:

$$H = \frac{\frac{1}{t_1} \sum_{i=1}^{n} \theta(i) \cdot w(L_i) \cdot L_i \cdot t_i}{\sum_{i=1}^{n} \theta(i) \cdot w(L_i)} \tag{3.5}$$

where n is number of images in the sequence, sorted by ascending time. The time t_1 is used to shift pixel values to the common time base. The function θ incorporates the results of ghost detection, where $\theta = 1$ for reference image and $\theta = 1 - G$ otherwise.

Chapter 4

Proposal of ghost-free HDR algorithm

This chapter contains the proposal of a novel ghost-free HDR merging algorithm, which is the core of my work during the pursuing of my Ph.D. The core of this chapter was published in Journal of Real-Time image processing as the article „De-Ghosted HDR Video Acquisition for Embedded Systems“ [39].

The scientific contribution of this thesis is the proof that:

A multi-exposure ghost-free HDR acquisition algorithm comparable to the state-of-the-art algorithms in quality can be designed for an embedded hardware device and achieves a real-time performance at high resolution.

The embedded hardware device should be based on FPGA technology with FullHD CMOS sensor onboard, at the same time be small in size and with low power demands to fit into the energy-efficient or battery-powered systems.

In this chapter, a novel architecture implementing the above idea in FPGA is proposed and its functionality and quality of output are experimentally proved. The chapter consists of the quality comparison to the related implementations and even state-of-the-art methods, that are too computationally demanding and even not feasible to implement and/or accelerate on FPGA. The aim is to show that proposed solution is simple, yet very powerful and providing good visual results at the same time. The performance and power consumption of algorithm implemented on various platforms is summarized at the end of this chapter.

The proposed novel ghost-free HDR acquisition method for stationary cameras is well implementable even in embedded systems in real-time with low resource requirements. While de-ghosting is being researched for a long time, the state-of-the-art methods that have good results are very computationally demanding and so they are not possible to implement in smart cameras and/or embedded systems attached to cameras. The novel HDR de-ghosting method proposed in this dissertation was designed with respect to real-time processing in embedded hardware and the low demands reflects positively even to performance of CPU implementation. [39].

Figure 4.1: Figure obtained from real application of proposed ghost-free algorithm - traffic monitoring system with licence plate detection, which demonstrates the contribution of proposed method. Top left - stripes of original images with a significant car motion. Top middle and top right - Images representing coefficients used for the HDR merging (*certainty maps*, see Section 4.1). Bottom left - ghosted HDR image. Bottom right - HDR image merged using proposed method.

4.1 Ghost-free merging algorithm

This section describes the proposal of a novel HDR merging method that produces ghost-free results. This approach is based on pixel value matching, the idea being similar to the solutions proposed by Grosch [15], Wu [66], and Wang [63] but with quite different and improved processing. The exposure time of each image is known; therefore, it is possible to estimate and match pixel values in the adjacent images, except for the over or under-exposed patches where the pixel values will obviously not match. Such estimation is not very precise, the captured image data is affected by factors such as noise, sensor quantization errors, CRF, etc. The reviewed methods generally use fixed or user-guided thresholds which must be employed in order to introduce user-defined tolerance to these factors. These fixed or user-defined thresholds often cause adverse effects in the final HDR images, such as visible transitions between static and motion areas etc. I propose a method to overcome such problems. [39]

4.1.1 Certainty map

In this approach, every image L_i is assigned a *Certainty map* C_i related to the reference image L_{ref}, which is generally considered to be the middle (exposure) image in the sequence. The *Certainty map* C contains values representing the estimated level of certainty that the individual pixels contain the same patch of the scene as the reference pixel, but obtained under a different exposure. Unlike ghostmaps, *Certainty maps* hold not only the patches containing motion, but rather all patches inappropriate for merging - such as under and over-exposed pixels. [39].

The probability distribution of low level value pixels is Poisson [29] due to the discrete nature of the incoming photons. With higher intensities, the distribution transforms into Normal (Gaussian). Therefore, I use the Gaussian function to derive the certainty (estimated probability) that the two luminance levels, estimated and measured, match. The *Certainty map* C_i (see Figure 4.2) replaces the binary *ghostmap* with soft assigned values, obtained using the information from the reference image L_{ref}, the estimated image $\overline{L_i}$, the exposure times t_i and t_{ref}, as well as the CRF. Note, please, that in this paper the inverse CRF was implicitly applied to all images L_i. Image $\overline{L_i}$ is estimated by the following equation:

$$\overline{L_i} = L_{ref} \cdot \left(\frac{t_i}{t_{ref}} \right) \tag{4.1}$$

Consequently, the estimated value for image i is processed along with the actual value of L_i to get the probability based *Certainty map* C_i as:

$$C_i = e^{-\frac{(L_i - \overline{L_i})^2}{2\sigma^2}} \tag{4.2}$$

where σ reflects the standard deviation of the pixel measurement (affecting the „softness“ weight). The lower σ is, the sharper or more strict the *Certainty* maps are, which results mainly in the dynamic range reduction. On the other hand, a high σ causes „softer“ *Certainty* maps, which may start to be ghosted. Ghost detection generally, and indeed inherently, cannot work well for the over and under-exposed spots of an image; thus the *Certainty* map algorithm contains a boundary condition: If the estimated value lies beyond the point of saturation, the Certainty is assigned at maximum value. [39].

Figure 4.2: Two *Certainty maps* (bottom) obtained from the sequence on the top. The *Certainty map* on the left was obtained from top left and top middle (reference) image, the *Certainty map* on the right was obtained from top middle (reference) and top right image.

4.1.2 Multi-exposure merging algorithm

Proposed modification of Debevec's [5] merging algorithm incorporates the weights from the *Certainty map*, obtained through Equation 4.2. The HDR image H is calculated as the weighted sum of pixels from n images using the following equation:

$$H = \frac{C_i \cdot w(L_i) \cdot L_i \cdot \frac{t_i}{t_{min}}}{\sum_{i=1}^{n} (C_i \cdot w(L_i))} \tag{4.3}$$

The C_i for reference image certainty is considered to be 1.

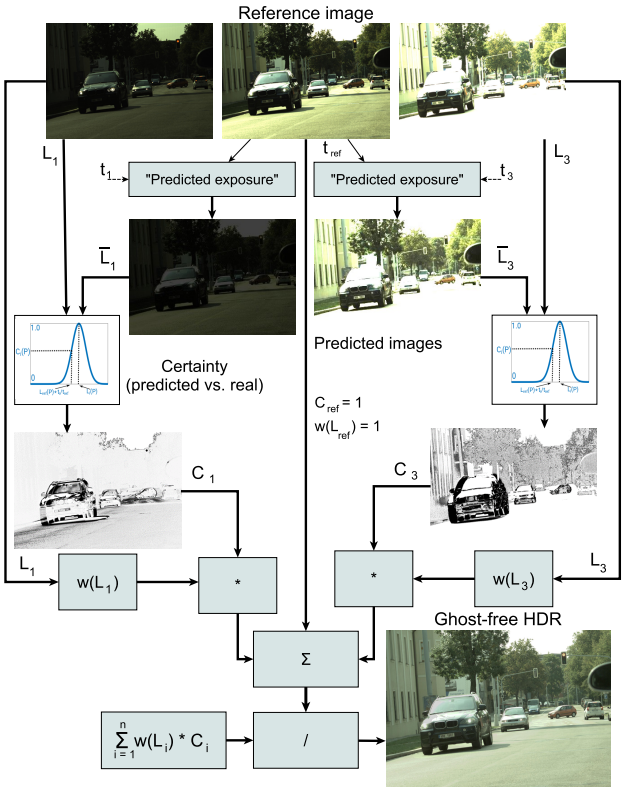


Figure 4.3: A scheme illustrating the proposed ghost-free merging of according to Equation 4.3 on a sequence of three images.

The $w(L_{ref})$ is considered to be 1, as the reference image is a „pattern“ with the desired object layout; it is not desirable to weight out the pixels, even if poorly exposed. A scheme illustrating the Equation 4.2 is shown in Figure 4.3. [39].

4.2 Implementation in HDR pipeline

This section describe how the proposed algorithm was implemented, with goal to incorporate it into processing pipeline of architecture by Nosko et al. [42].

The proposed algorithm was designed, in line with the previous assessment, with respect to real-time processing using embedded hardware – we are considering mainly FPGA based platforms and SoC equipped with GPU (e.g. NVIDIA Tegra). The standard desktop CPU implementation is included mainly for comparison.

The ghost-free merging unit consists of two components, *Certainty* map creation (Section 4.1.1) and HDR merging (Section 4.1.2). The *Certainty* map is obtained by predicting and matching the luminance levels, thus it is necessary to provide luminance images. The RAW data from sensors, e.g. in embedded devices/cameras, should, therefore, to be converted to luminance because if the individual RGB channels are processed separately, their saturation, which is independent for each channel, may lead to results of *Certainty* different for individual color channels and thus to adverse color shifts during HDR merging.

The value of σ should be adjusted based on the image sensor noise, including the quantisation noise, and the Exposure Value (EV) step (exposure time ratio) between the

individual images in the HDR sequence. In our reference implementation, we achieved the best results with $\sigma = 5$ for the exposure step of 1EV and $\sigma = 11$ for 2EV.

The proposed merging algorithm (Section 4.1.2) is applicable on grayscale, RGB and also RAW image data. The RAW data can be merged in the original form before debayering and debayered afterwards, as proposed by Tamburrino et al. [55]; thus, this implementation can save approximately 2/3 of the operations comparing to merging in RGB space.

The proposed ghost-free merging is applicable to an arbitrary number of images in the sequence of exposures. However, the following implementation and performance comparisons are related to merging of three images, unless stated otherwise. [39].

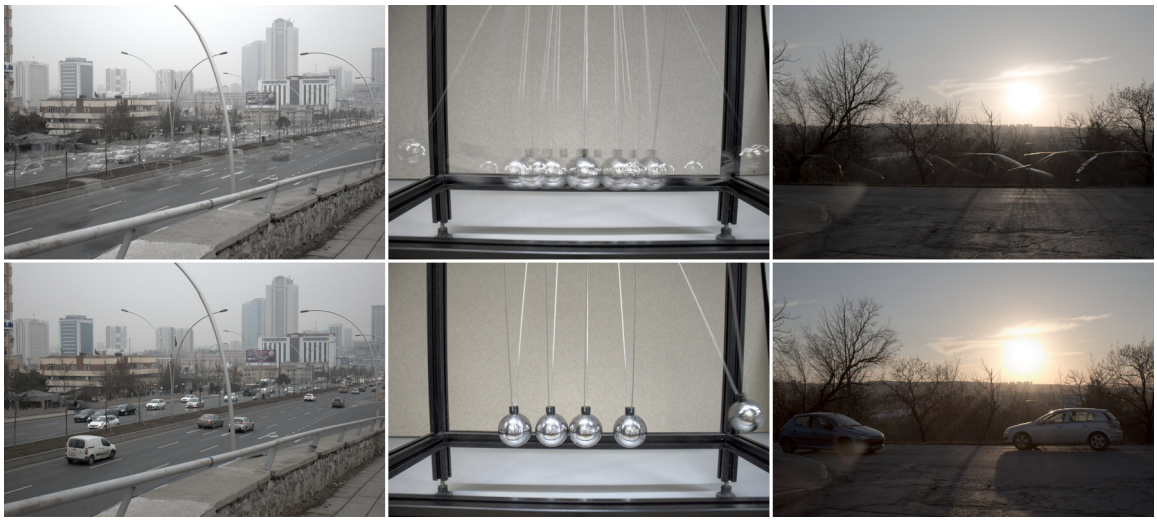


Figure 4.4: Ghosted HDRs (top line) and HDRs merged using proposed ghost-free method (bottom line) on sequences „Fast cars“ [59] (left), „105“ [60] (middle) and „117“ [60] (right). Datasets contains 9 LDR (Low Dynamic Range) images.

The proposed algorithm was implemented as a part of FPGA based HDR video acquisition pipeline[42]. The pipeline was implemented on platform equipped by SoC Xilinx Zynq XC7Z020 (see Figure 3.24).

Computational optimizations

The proposed algorithm performs per-pixel processing and requires a relatively small number of per-pixel operations. Some of its functionality is computationally demanding (e.g. division and Gauss function calculation), however, it can be optimised and/or tabulated. Inverse CRF and triangle weight functions can be tabulated thanks to the limited number of possible LDR (Low Dynamic Range) pixel values. The ratio between exposures t_i and t_{ref} in Equation 4.1 can be calculated once for each setting of LDR exposure times. The Gaussian function (Equation 4.2) can be convenient because the pixel values are discrete and only a finite combination of pixel values is possible, especially when considering only the differences between the captured and predicted values. The number of the Gaussian function results with relevant certainty, e.g. $> 1\%$ is limited, especially for a higher σ. The evaluation in Section 4.3 is performed with $\sigma = 11.0$, which leads only to 35 various results.

The functions represented in the tables are pre-calculated using the processors present in the embedded acceleration platforms. If needed, they can be updated while the accelerator executes the main algorithm.

Figure 4.5: Output of the proposed ghost-free merging method on the sequence of Gallo [14] (top). Previews of the various algorithm results are shown at the bottom: Gallo et al. [14] (A), Jacobs et al. [18] (B), Pece et al. [44] (C), Zhang et al. [71] (D) and proposed algorithm (E). The previews A to D are published online at http://www.vsislab.com/projects/IPM/HDR/project.html.

Evaluation of precision

The CPU and GPU reference implementations are written in C++ and CUDA, using standard 32-bit floating point data type. The whole FPGA design is implemented using only fixed point data representation and arithmetic, which is natural and also efficient for FPGA hardware. The ranges of numerical values in the individual pipeline stages are known; therefore, it is feasible to adapt the bit width of the individual parts of the pipeline to achieve a sufficient range (and precision) without using the floating point representation, whose resource requirements are generally much higher. The FPGA implementation is fixed for merging three LDR images with up to 10 bit depth. The input of the ghost detection block consists in three corresponding pixels in 10.8 fixed point representation (10 bits for integer and 8 bits for decimal part). The fractional part can be used for the data after the linearisation process (application of CRF – Camera Response Function [5, 48, 37]). The resulting *Certainty maps* are in the 1.10 format and all further mathematical operations during the HDR merging are performed using 10.12 precision. The accuracy of fixed point arithmetic comparing to the software float implementation is evaluated using PSNR and MSSIM metrics. The ghost detection and merging achieved PSNR of 51.1 and 58, MSSIM is over 99% for both algorithms, using the above mentioned 12bit fractional bits.

The Table 4.1 presents an FPGA resource consumption of proposed design. The abbreviations in the table describes the FPGA primitives: LUT – Look-up Table; FF – registers;

Table 4.1: FPGA Resource utilisation for merging 3 LDR images of 1920 × 1080 pixels. Design is routed for Xilinx Zynq Z-7020.

	LUT	LUTRAM	FF	BRAM	DSP
Certainty maps	3532	–	3339	4	4
HDR merging	893	–	2570	10	16
Total (HLS)	4425	–	5909	14	20
Total (Routing)	**1057**	**252**	**2052**	**2**	**16**
available	53200	17400	106400	280	220
utilisation [%]	1.99	1.45	1.93	0.72	7.27

BRAM – Block RAM (36kbit block of distributed memory); DSP – Digital Signal Processing block (used as a multiplier); LUTRAM – LUT-based small distributed memory.

A line „Total (HLS)“ indicates the amount of resources estimated by Xilinx High Level Synthesis (HLS) design tool[1]. Such resources are quite often overrated and the Place and Route process optimises out an unnecessary logic (see line „Total (Routing)“) for the target FPGA. The Table 4.1 shows e.g. most of BRAM resources were conveniently converted into LUTRAM, probably due to only a few Gauss coefficients needed to store, as explained in Subsection 4.2.

Table 4.2: Resource utilization of complete camera solution of Nosko et al. [42] enhanced by the proposed ghost-free merging block, comparing to Bouderbane [3].

	LUT	LUTRAM	FF	BRAM	DSP
Prop. pipeline	39145	3137	53592	51	58
Bouderbane [3]	49193	–	50399	35	20

The proposed algorithm was implemented into FPGA based HDR video acquisition pipeline proposed by Nosko et al.[42]. The proposed algorithm was designed to replace the original and very simple „Deghosting & merging“ block (please refer to Nosko et al. [42]). The Table 4.2 compares the resources consumed by such pipeline with pipeline from Bouderbane et al. [3]; unfortunately, they do not provide more detailed statistics. For detailed description regarding pipeline, please refer to the article by Nosko et al. [42]. Please note that proposed design is built on Xilinx Zynq and Bouderbane camera on Virtex-6 and also that in Nosko's pipeline, more than 1/3 of LUT and Register resources and most of BRAM and DSPs are occupied by local tone-mapping operator [42].

4.3 State-of-the-art ghost removal evaluation

This section is focused on evaluation of proposed ghost-free HDR merging and provides the comparison to the state-of-the-art algorithms.

The proposed algorithm is evaluated on HDR datasets focused on evaluation of HDR deghosting methods [60, 59, 23], on the image sets retrieved from related articles [14, 50, 21] and also on the image sets captured by camera prototype by Nosko et al. [42] (see Section 3.4).

[1] www.xilinx.com

Figure 4.6: Sample outputs of related deghosting algorithms by Pece et al. [44] (left) and Min et al. [35] (right) on the scene from Figure 4.1. Our experiments revealed that listed algorithms should be successful only on images with convenient histogram distribution.

The results of the proposed ghost-free merging are presented in Figures 4.1, 4.4, 4.5, 4.9, 4.8, 4.10 and 4.12. Our method is suitable for almost any application with stationary cameras. Besides the evaluation of various generic datasets, the ghost removing capability was evaluated on a traffic monitoring task, where the main goal was to preserve the greatest possible level of detail so that the images can serve as evidence, with the readability of the licence plates of the vehicles in motion playing the most important part. Figure 4.1 contains a car approaching camera at approximately 50km/h. Still, six exposures ($\sim$ 66ms at 90FPS) were intentionally omitted between the images to show the capability of the ghost removing for e.g. faster moving objects.

According to presented results, the visual outputs are comparable to the state-of-the-art; however, the proposed algorithm is capable of running in real-time, while state-of-the-art algorithms require long offline processing in terms of seconds or even minutes per image.

The experiments with the related and state-of-the-art algorithms discovered that most of the de-ghosting methods related to our approach are very dependent on scene composition, luminance distribution, or other assumptions. Proposed approach does not have such limitations, it is more robust, and does not require user-guided tuning of parameters, unlike algorithms with similar complexity.

Probably only related work, which implements any ghost-free merging on embedded device, in this case on FPGA, was proposed by Bouderbane et al. [4]. They use method by Debevec and Malik [5], where they combined weighting function from Debevec with weight function proposed in their previous paper [3]; their method was inspired by the work of Sidibe et al. [51]. However, the ghost detection is based only on weak assumption, as Bouderbane use the weight function, which gives a higher factor for pixels whose recovered radiance value are closed to the recovered radiance of reference values and low factor for pixels whose radiance values diverge considerably from pixels radiance value of the reference image. The evaluation of deghosting quality is rather limited, as source code of their reference solution is not available and they did not evaluate their algorithm on any third-party dataset. On the image data supplied within the article, the method suppress ghosting quite well(see Figure 4.7) but the slight ghosting effect is still present (see results in the article [3]), also the dynamic range is quite reduced, even in parts with static background.

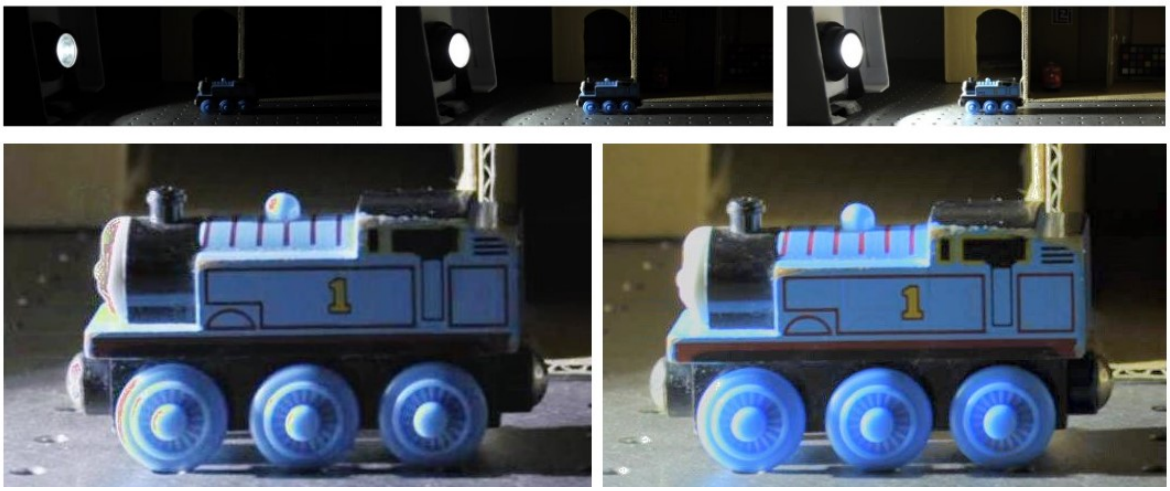


Figure 4.7: Figure shows the ghost-free HDR outputs of Bouderbane [4] (left) and proposed method (right), both tonemapped by Duan [8] operator. Bouderbane result and source images are retrieved from [4]. Please mind the color shift in very bright patches of Bouderbane result.

Algorithm by Gallo et al. [14] operates on relatively large rectangular patches (e.g. 40x40 pixels [14]) instead of individual pixels. If the patch contains large number of pixels not corresponding to patch from the reference image, the patch is omitted from merging. As the patches used in the algorithm are quite large, visible artifacts occur at their boundaries; the authors suggest their suppression by Poisson blending.

The methods based on histograms [44, 35] have a common issue, the scene has to be balanced from the point of histogram equalization. The method presented by Pece et al. [44] is marking pixels as ghosts based on decision, whether the pixel changes its relative position in histograms over all of the expositions. The position in histogram is acquired by comparison with median pixel value. If median is very low/high, for example if the scene has large large number of under/overexposed patches, the change of pixel position in histogram cannot be reliably detected. In the method proposed by Min et al. [35], one median threshold is replaced by eight percentiles and whole histogram is divided into nine segments with equal number of pixels, but it only mitigates the same issue. The example outputs of the Pece et al. [44] and Min et al. [35] algorithms on data obtained by Nosko's camera [42] are shown on Figure 4.6.

In general, the existing methods are more or less using fixed or user-adjusted thresholds and binary ghost maps, which either includes the pixel into the merging process or omits it completely. Such approach negatively affects the merging process and appearance of the resulting HDR image, causes higher noise on the affected patches around the moving objects, and also on wrongly detected patches.

4.3.1 Dataset evaluation and comparison

I performed the evaluation on datasets [23, 59, 60], containing sequences of images of various scenes and different types of motion. The results provide a comparison of the proposed method with generally more precise and computationally demanding methods, commonly based on optical flow, which were not even included into the related work due to their complexity and high computational demands.

Figure 4.8: The source sequence (top left) is merged with (bottom) and without (top right) proposed ghost-free merging algorithm. Source images retrieved from Sing Bing Kang [21].

Figure 4.9: Output of the proposed HDR ghost-free merging method for *Complex Scene 1* of dataset [23] (left). Ghosted HDR image is shown on the right. Previews of various algorithm results are shown at the bottom. No de-ghosting (A), Silk et al. [52] (B), Sen et al. [50] (C), Photoshop (D), Photomatix (E) and proposed algorithm (F). The previews A to E are published as a part of a Karaduzovic dataset [23].

One of the datasets [23] contains multiple scenes with artificial objects movements. Its advantage consists in the existence of the ground truth image, which allows a comparison to the results as well as to many results of various published methods [17, 50, 52]. Figures 4.5 and 4.9 show the capabilities of the proposed method, showing that it provides results visually comparable to optical flow based methods.

Tursun et al. [59, 60] published two datasets and proposed metrics for evaluation of HDR de-ghosting quality. The evaluated samples from the datasets are shown in Figure 4.4 and the HDR quality metric [59] is evaluated in Table 4.3. The metric evaluates the dynamic range achieved inside the motion regions, considering also the correctness of the de-ghosting. The image sets, in which we got worse results than other algorithms, were successfully de-ghosted anyway; however, the worse results were probably caused by losses in the dynamic

Table 4.3: Results of the „Dynamic Region Dynamic Range“ metric proposed by Tursun [59] and evaluated on their dataset. The metric evaluates the resulting dynamic range within regions containing movement; the higher the value, the better.

Metric „DR“	[15]	[50]	[52]	none	**This work**
Cafe	**2.63**	2.61	2.60	2.47	2.42
FastCars	1.12	1.18	1.10	1.10	**1.38**
Flag	1.40	1.50	1.49	1.45	**1.59**
Gallery1	1.59	1.59	1.56	1.55	**1.70**
Gallery2	2.41	**2.56**	2.14	2.29	2.05
LibrarySide	1.78	1.93	1.60	1.76	**3.20**
Shop1	2.20	2.39	2.00	2.10	**2.42**
Shop2	2.68	2.72	**2.89**	2.55	2.42
WalkingP.	1.94	**2.07**	1.83	2.05	1.58

Table 4.4: Evaluation of the HDR-VDP2 [33] metric on a „complex“ scene from Karaduzovic's [23] dataset.

	scene1	scene2	scene3	scene 4
Q	73.24	76.20	82.83	71.58

range. Evaluation of the proposed method on these datasets also proves that the proposed method is generally usable for sequences larger than two/three images, commonly used in cameras. In all the referenced datasets [23, 59, 60], the proposed algorithms achieved results visually comparable or even better than more complex algorithms (see Figure 4.10). However, the proposed method and also many HDR de-ghosting methods may yield artifacts in regions where the moving objects in the reference image are poorly-exposed, as Tursun et al. concluded [59].

Another metric that have been found useful is HDR-VDP2 by Mantiuk et al. [33]. The metric evaluates the visibility and quality differences in image pairs and represents a probability that an average observer will notice a difference in the images in the pair (see Figure 4.11). The essential problem for the metric evaluation is the absence of ground truth images. Applying this metric on image sets without ground truth reference seems useless, as even the state-of-the-art algorithms may fail in ghost detection and/or changes in the image quality e.g. by bluring of motion regions (see top of Figure 4.10). As a result, the metric output obtained on such data does not have any meaningful value.

Karaduzovic's [23] dataset contains ground truth images, because it contains scenes with artificial object motion. The metric was evaluated on „complex“ scenes and used the HDR merged from the ground truth sequence as a reference. The ground truth sequence is processed also by our algorithm (with de-ghosting disabled) to eliminate the effect of unrelated image enhancements and enables the direct comparison of the resulting HDR images. Table 4.4 contains an overall „quality“ metric of the produced ghost-free HDR output according to HDR-VDP2 [33] metric. Figure 4.11 shows „scene 1“ with highlighted differences between ground truth HDR and ghost-free HDR.

Figure 4.10: Figure shows scene „Cafe“ from Tursun's [59] dataset processed with Sen [50](top) and the proposed ghost-free algorithm(bottom). Sen [50] produce a heavily blured image, which precludes the HDR-VDP metric [33]; moreover, the de-ghosting method fails (see marked areas, where objects are shadowed and blured).

4.4 Performance summary

The performance of the algorithm on the relevant platforms is summarised in Table 4.5. Only the core parts, the certainty map creation and HDR merging were benchmarked, without including any data preprocessing time – please assume that at least in the FPGA and GPU implementations, the images are transferred into the memory using DMA in the background, without any performance losses. With the proposed optimisations, the algorithm is single-pass only. Table 4.5 compares the performance of the proposed Ghost-free merging of three LDR images on FPGA, SoC GPU and CPU platforms. In the case of FPGA, the design achieves target frequency of 200MHz and is fully pipelined; therefore, it allows the production of result pixels every clock cycle. Unlike in the sequential CPU and GPU processing, increasing the amount of work that the FPGA pipeline performs leads to consumption of more resources and prolonging the processing pipeline, which has a negative influence on latency; however, the data throughput remains the same (see Table 4.5).

Table 4.5: The table compares the performance of the proposed ghost-free merging of 3 LDR images (Figure 4.2) with a resolution of 1920×1080 on following platforms: FPGA Xilinx Zynq, embedded CPU and GPU Nvidia Tegra TX2 and CPU Intel Core i7-3770 (single core).

	FPGA	TX2 GPU	TX2 CPU	CPU
Certainty map [ms]	10.3	1.59	45.9	16.6
Merging [ms]	10.3	4.58	112.3	23.0
Total [ms]	10.3	6.17	158.2	39.6
Overall FPS	96.45	162.07	6.32	25.25

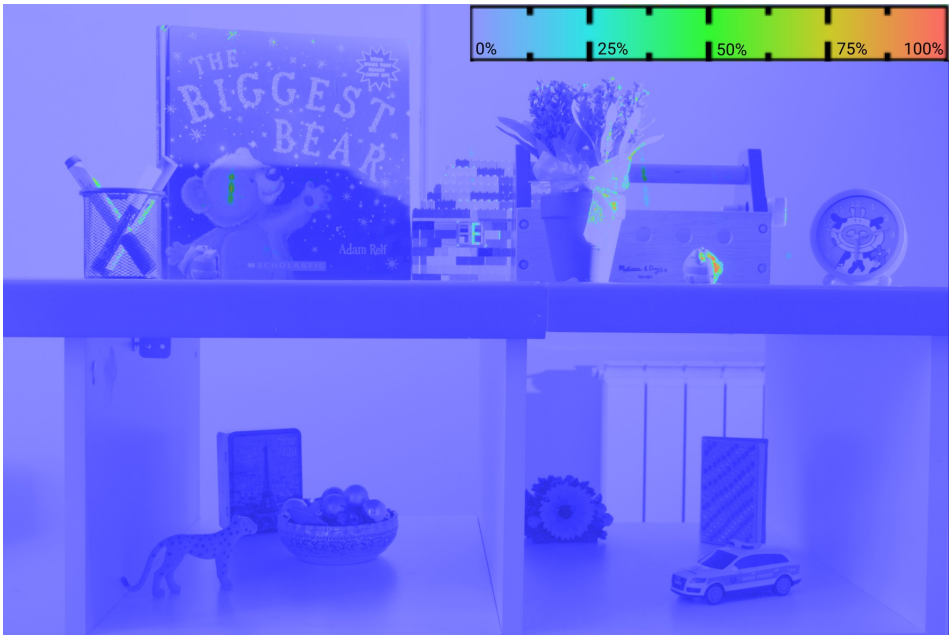


Figure 4.11: Figure shows „scene 1“ from Karaduzovic's [23] dataset processed by the HDR-VDP2 [33] metric. The colour bar reflects the probability that an average observer will notice a difference between ghost-free HDR and ground truth HDR. De-ghosted HDR visual quality, according to HDR-VDP2 metric [33] is 73.24 (see Table 4.4).

I chose the HDR camera prototype by Nosko et al. [42] for the integration of the proposed ghost-free method. I chose this camera prototype due to its compact size, presence of a FullHD resolution CMOS (and optionally with even higher resolution) and presence of Xilinx Zynq SoC. Moreover, I participated on the development of Nosko's prototype as well. The proposed method and its implementation into architecture Nosko et al. [42] have not been published yet; however, the performance parameters are already known. Moreover, I have designed the new ghost-free merging block as a 1 to 1 replacement of previously published HDR merging with ghost removal[42], then the overall design shares all other features, such as advanced local tonemapping.

The proposed algorithm should also be easily integrated into existing solutions of HDR acquisition devices by Popadic et al. [45], Lapray [27], Nosko [41] and probably others which are all based on pixel weighting, similar to Debevec and Malik [5].

The implementation on Nosko's platform allowed direct comparison with other architectures, however, probably only related work, which implements multi-exposure ghost-free HDR acquisition on an embedded device, in this case on FPGA, was proposed by Bouderbane et al. [4]. The Table 4.6 provides an overall comparison to Bouderbane solution.

As can be seen on Table 4.6, the design outperforms the Bouderbane architecture in all parameters. The design is fully pipelined, producing HDR pixel in every clock cycle. Target clocking frequency is 200MHz, which enables the acquisition of FullHD HDR images at up to 96FPS. The fixed point arithmetic has a positive contribution to clocking frequency, low resource requirement and low power consumption. At the same time, the calculations are still performed in high accuracy, which was summarized in Section 4.2. The platforms are both implemented in different FPGA family, where Zynq is part of 7th and

Figure 4.12: A car passing by the camera – ghosted HDR (left) and result of the proposed ghost-free merging algorithm (right).

Table 4.6: Comparison of main parameters of proposed solution to Bouderbane et al. [4].

	Proposed pipeline on [42]	Bouderbane
Platform	Zynq 7020	Virtex 6
Resolution	1920×1080	1280×1024
TMO	Durand (Local)	Duan (Global)
Arithmetic	Fixed point	Floating point
Maximum speed	$200Mhz$	$114.2Mhz$
Throughput	$200Mpix/s$	$114.2Mpix/s$
Framerate	$96FPS$	$60FPS$

Virtex part of 6th series from Xilinx. However, Virtex is a High end, while Zynq(Artix) only mid or low-end FPGA. The Table 4.2 shows the overall FPGA resource consumption for both complete camera solutions; Bouderbane does not provide separately the resources consumed by deghosting and HDR merging circuits. Please notice that local tonemapping operator with bilateral filter (on Nosko's platform) require more than 1/3 of overall LUT and Register resources and consumes most of BRAM and DSP resources.

The following part of the evaluation aims to compare proposed algorithm performance to related state-of-the-art implementations. The essential problem is those relevant algorithms are not generally available in the form of code or executable; therefore the performance comparison is rather limited to algorithms, where I managed to get source codes to run or where I get required information from relevant articles.

The comparison from Table 4.7 confirms that deghosting algorithms do not generally achieve real-time performance. Depending on the algorithm and desired deghosting quality (if available), the process can take from tens of seconds to more than ten minutes. Certain algorithms, such as Grosh [15] which have similar computation complexity as the proposed algorithm, do achieve relatively low processing time, however, the output is not deghosted very well, as shown in evaluation by Tursun et al. [59]. The Table 4.7 shows, that proposed algorithm is running much faster than any compared algorithm and is even twice faster than the algorithm by Grosch [15]. Please note, that average times from Table 4.7 are valid for sets of nine images with 4MPix resolution.

Table 4.7: Table results obtained from Tursun et al. [59]. The table shows the average processing time of deghosting algorithms on Tursun dataset [59]. Source image sets contain 9 images with 4MPix resolution (Sen and Khan merges three images with 1024×683 only). Tursun achieved these results on CPU Intel i7-3770; however, it is not specified whether the algorithms utilized all CPU cores or not. Proposed method result is benchmarked as single-core.

	Proposed	Grosch[15]	Khan[24]	Sen[50]	Silk[52]	Hu[17]	Tursun[59]
Avg. time[s]	0.48	1.04	616.45	209.78	14.33	230.36	7.09

The Table 4.8 compares the performance of proposed algorithm and algorithms by Pece et al. [44] and Min et al. [35]. I chose these algorithms to compare due to its possible easy implementation on FPGA; the author claims that the algorithm does not use the multiplication, division, and floating-point operations for object motion detection. Moreover, all operations, including the histogram calculation, are relatively easily implementable on FPGA. Also, the deghosting ability presented in the paper seemed to be very promising.

Table 4.8: Table compares the performance of the proposed algorithm with algorithms by Pece et al. [44] and Min et al. [35]. Algorithms were benchmarked on CPU Intel i7-3770 (single thread) on a scene from Figure 4.1. Source image set contains three images with FullHD resolution (1920×1080).

	Proposed	Pece et al. [44]	Min et al. [35]
Avg. time [ms]	39.6	193	206

I implemented the proposed algorithm and algorithms by Pece et al. [44] and Min et al. [35] in C++, so the performance presented in Table 4.8 should be comparable. As could be observed, the proposed algorithm is almost five-times faster. Moreover, the deghosting results are also better, as can be observed in Figure 4.6. The explanation of why these methods do not achieve good deghosting results is in the Subsection 4.3.

The Table 4.9 presents the performance results obtained by Yan et al. [69] and presented in their article. They compared a number of algorithms and benchmarked them on CPU Intel i7 and GPU NVIDIA GeForce GTX 1080Ti. As can be observed, the proposed algorithm achieved better performance on CPU architecture (single-core) than others on even high-end GPUs. Yan's [69] and Wu's [67] CNN-based merging are relatively fast; however, they run on high-end GPU, which consumes much more energy than CPU (up to 280W).

Table 4.9: Table results obtained from Yan et al. [69]. Table shows average processing time of deghosting algorithms on three images with resolution 1000×1500. CPU used is Intel i7 (not further specified by Yan), GPU used is NVIDIA GeForce GTX 1080Ti.

Algorithm	Proposed	Yan [69]	Wu [67]	Kalantari [20]	Sen [50]	Wu [66]
Platform	CPU	GPU	GPU	CPU+GPU	CPU	CPU
Time [s]	0.022	0.31	0.24	29.14	61.81	79.77

4.4.1 Power consumption summary

This subsection provides a brief comparison of the power consumption of selected algorithms. The overall energy in Joules per one frame is estimated from processing time and TDP of a processor; alternatively, FPGA consumption is estimated by the design tools based on the amount of logic used. The Table 4.10 and Table 4.11 contains the energy requirements converted from performance summary in Table 4.7 and Table 4.8.

Table 4.10 presents that the proposed algorithm has the lowest power consumption among all of the measured algorithms. The results were achieved on the dataset published by Tursun[59], which contains sequences of nine images with 4MPix resolution. It is not specified by Tursun [59] whether the algorithms utilize single or multiple cores; therefore, I assume only single-core implementation as a lower estimate of the possible power consumption. Proposed method result is benchmarked as single-core.

Table 4.10: Table of energy consumption per HDR frame, derived from Table 4.7. The table shows average energy consumption for processing one HDR frame of deghosting algorithms on Tursun dataset [59].

	Proposed	Grosch[15]	Khan[24]	Sen[50]	Silk[52]	Hu[17]	Tursun[59]
Avg. energy [J]	12	26	15411	5244	358	5759	177

Table 4.11: Table compares the energy consumption for processing one HDR frame by proposed algorithm with algorithms by Pece et al. [44] and Min et al. [35]. Algorithms were benchmarked on CPU Intel i7-3770 (single thread) on scene from Figure 4.1. Source image set contains three images with FullHD resolution (1920×1080).

	Proposed	Pece et al. [44]	Min et al. [35]
Avg. energy [J]	0.99	4.82	5.15

The average consumption is 12J per one HDR frame, which is 46% of the second least demanding algorithm by Grosh [15]. Moreover, proposed algorithm demands are measured for single-core processing only, whether the data provided by Tursun [59] are not specified whether were achieved on single-core only; however, the results in Table 4.7 and Table 4.10 assumes they are.

Table 4.12: The table compares the power consumption of the proposed algorithm on the CPU and FPGA platform and shows the estimated energy required for the ghost-free merge of the sequence of three FullHD images. Please note that consumption of Camera Nosko [42] includes camera as a whole.

	Consumption [W]	Energy per frame [J]	comp. to CPU[%]
CPU Intel i7-3770	25W (single core)	0.99	—
Camera Nosko [42] (30FPS)	∼8W	0.266	26.9
Camera Nosko [42] (96 FPS)	∼8W	0.083	8.3
Tegra TX2 - GPU only	15W	0.093	9.3
Proposed - FPGA only	1,1W	0.0115	1.16

Table 4.13: The table compares performance of proposed Ghost-free merging of three LDR images (Figure 4.2) of resolution 1920 × 1080 on FPGA and CPU platforms. Data are selected from Table 4.5.

	FPGA Xilinx Zynq	CPU Intel Core i7-3770
Ghost det. [ms]	10.3	16.6
Merging [ms]	10.3	23.0
Total [ms]	10.3	39.6
Overall FPS	96.45	25.25

The HDR camera by Zemcik et al. [70] achieved overall power consumption of 12W and the HDR cameras by Nosko et al. [41, 42] even less, total 8W. Based on the performance summarized in Table 4.5 and assuming the maximum speed of 96.4FPS, the camera Nosko et al. [42] with proposed algorithm consumes $0,083J$ per frame. The power consumption of CPU Intel Core i7-3770 was measured in single-core load (running proposed algorithm) and achieved 25W. CPU achieved framerate of 25.25 FPS, which results in consumption approximately $0.99J$ per frame; note, please, that the difference between standby and the full load was power consumption measured, which shows only the desired dynamic part of power consumption.

In summary, the HDR camera by Nosko et al [42] with proposed algorithm consumes only 8.4% comparing to the CPU implementation. Moreover, most of the power consumption of HDR camera is spent on camera hardware, including CMOS chip and H.264 encoder, while the consumption of the FPGA itself consumes approx. $1,1W$ only (estimation by Xilinx Vivado tool). This result is much more favourable for FPGA, but the comparison is fairer because it compares only the „computing“ elements. The energy spent on one frame drops to approx. $0.011J$, which is little above 1% of the energy consumed by CPU.

4.5 Validation and scientific contribution

at the beginning of this Chapter 4 it was stated that the scientific contribution of this thesis should be the proof of the following hypothesis: *A multi-exposure ghost-free HDR acquisition algorithm comparable to the state-of-the-art algorithms in quality can be designed for an embedded hardware device and achieves a real-time performance at high resolution.*

In Section 4.1, I proposed a **Ghost-free HDR** acquisition algorithm implementable on **FPGA**. This method was implemented and its description is included in Section 4.2. The proposed Ghost-free algorithm produces a visual output **comparable to the State-of-the-art** as evaluated in Section 4.3. Finally, the proposed design achieves more than **real-time performance** of 96FPS on fullHD resolution as summarized in Section 4.4. Therefore, I consider the hypotheses validated.

In more detail, the proposed novel ghost-free HDR merging algorithm is suitable for real-time implementation in embedded devices. The algorithm is well suitable for implementation on many platforms, including the CPU and GPU based platforms. However, the aim of contribution was a successful implementation of such an algorithm into FPGA, which was experimentally proved in Section 4.2. Also, the target performance, which is real-time processing on FullHD resolution, was fulfilled, since the proposed solution is able to run on up to 96 FPS (Table 4.5). At the same time, the proposed solution outperforms

the FPGA solution of only state-of-the-art FPGA implementation of Bouderbane et al. [4], which achieved only 60 FPS on HD resolution (1280×1024).

The performance comparison with most of the state-of-the-art algorithms requires a CPU reference implementation. The performance evaluation in Section 4.4 shows that the algorithm performs well even on CPU; single-core implementation achieves up to 25.25FPS, as shown in Table 4.5, which is the best result. The second least demanding state-of-the-art algorithm, according to Table 4.7 is from Grosh [15]. Under the same conditions and on the same dataset[59], the proposed algorithm is faster by approx. 54%. Table 4.9 further compares the proposed algorithm with the latest and GPU accelerated state-of-the-art algorithms. Finally, Table 4.11 compares the CPU performance of related algorithms which I reviewed to be suitable for FPGA implementation. The proposed algorithm achieved the best result and is 4.8 times faster than the fastest FPGA implementable algorithm.

Table 4.7 and Table 4.11 compares the CPU power consumption to state-of-the-art algorithms and, linearly with performance, requires only 46% of power comparing to the second least demanding algorithm by Grosh [15].

The CPU implementation itself is so fast that almost accomplished the real-time requirement; however, the real benefits of the method stand out along with FPGA acceleration, which fundamentally affects the performance and power consumption. The Table 4.12 and Table 4.13 shows the effectiveness and benefits of FPGA acceleration of proposed algorithm. While the FPGA implementation offers almost 4-times higher performance comparing to CPU (25.25 FPS) and reaches the 96FPS, the energy consumption drops by 98,86% per frame. These parameters should be even much better in case of ASIC chip production (or integration into an existing chip, e.g. as an accelerator block), for which the FPGA reference implementation is necessary. However, I did not have such funding and contacts to ASIC manufacturing facility.

The comparison to the state-of-the-art algorithms (Section 4.3) and evaluation of HDR datasets (Section 4.3.1) shows, that proposed algorithm is performing ghost-free HDR merging well and the ghost effect is removed, at the same time have better results and is much more robust than related algorithms. The results are even comparable to the state-of-the-art optical flow-based algorithm, which belongs to the class of performance demanding, offline processing algorithms.

4.6 Applications and future work

The usability of proposed work is relatively broad and involves almost every scenario, where some camera is used and where it may appear difficult lightning conditions. From the possible applications, I focused mainly on surveillance, security and traffic monitoring systems where the HDR video capture significantly improves the reliability of systems under bad weather or light conditions. Here I present several applications, where the proposed algorithm and architecture for ghost-free HDR acquisition can take place and should improve the system utility value.

Figure 4.13: A HDR scene captured by HDR camera prototype by Nosko et al. [42].

Road and traffic monitoring

During the day, the outdoor lighting conditions change a lot, and the traditional LDR cameras cannot capture the scene very well. Such bad conditions include shadows, flares, direct sunshine or night, etc. During the day, from the sunset to sunrise, the angle of sunshine changes and quite often disables the ability of the camera to expose a good looking image. During the night, the reflexive elements often over-expose a certain part of the scene, which is then unreadable.

Another, however similar situation happens when we try to observe a scene, where the large luminance differences already are, for example, insight into/from tunnels, subways, building's entry and many other. The auto-exposure algorithm can adjust exposition time to improve the result; however, it is often driven by image statistics such as image histogram, which does not always take into account the desired/monitored part or objects in the scene.

The proposed algorithm should help to capture, in the best case, all range of luminance in the scene and therefore provide much more image details for further processing or evidence purposes. Figure 4.14 shows the output of the overview camera, where the HDR mode allowed to capture details, which would be lost with the standard camera (vertical road signs, the interior of the trucks etc).

Section speed enforcement

The section speed enforcement composes from two portals with road traffic cameras. The system monitors vehicles going through spots on the road and matches the records using license plate reading (ANPR/ALPR) software. This enables to determine the average speed in the entire road section measuring from hundreds of meters to tens of kilometres. Such systems are often deployed on the spot, where the lighting conditions are not ideal for standard cameras, e.g. at the entrance and the end of tunnels, on direct sunlight etc.

Figure 4.14: A Graphic output of HDR camera prototype from Nosko et al. [42]. The image is tonemapped by Reinhard et al. [47] operator. The source image sequence is shown in the right column.

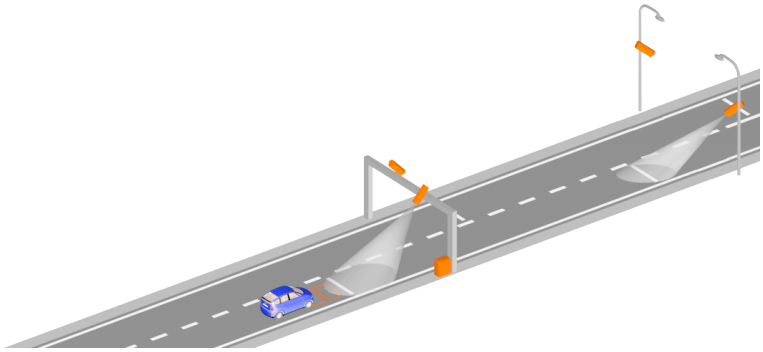


Figure 4.15: A scheme of section speed enforcement. Image obtained from *www.camea.cz*.

Figure 4.16: A real photos taken from section speed cameras (without HDR mode). The actual position of the sun makes the licence plate unreadable. Images obtained from Camea spol. s r.o. (*www.camea.cz*).

A critical part of the system is a successful license plate detection on both portals and also capturing of evidence photo, where the essential is to capture visible license plate, a whole car with details and ideally even the driver. Typically, there are large differences in luminance, since the driver's face is often hidden in the shadow and license place is white, provided with a reflective layer. Also the actual lightning and weather condition may affect the license plate detection – Figure 4.16 shows the real photos from section speed cameras (see Figure 4.15). The actual position of the sun creates a shadow which disables the successful license plate reading. HDR mode with a proposed ghost-free extension would significantly improve the licence plate detection rate (as the cars are captured in motion).

Inexpensive units for surveillance

Proposed ghost-free HDR merging is embeddable into smart FPGA-based devices and therefore, can take place in small, embedded devices focused on surveillance. It can also extend existing devices, if they are based on FPGAs, due to its small resource requirements. Moreover, from FPGA implementation is only a small step to deploy HDR core into a specialized custom chip or as an accelerator into ARM processors. Figure 4.17 and Figure 4.18 illustrates the benefits of ghost-free HDR acquisition on car onboard cameras, which, besides the surveillance purposes, can play a significant role in autonomous vehicles. Moreover, with the HDR image compression published in Zemcik et al. [70] it is possible to record the HDR video using standard H.264 encoder.

Figure 4.17: Illustrative comparison of non-HDR mode (left) and HDR (right) on an automotive application. Image was taken by Hyperyon camera and obtained from https://www.e-consystems.com.

Surveillance systems and crowd analysis

The surveillance systems can also profit from ghost-free HDR merging. Depending on the place of monitoring, the demands for such systems vary a lot. When monitoring outdoors, we can face the same weather problems as in road traffic monitoring task. The Figure 4.19 contains an example situation in surveillance applications – the background is well-exposed; however, objects of interest are poorly visible. This is caused by a large difference of luminance in the background and of the people/objects. Standard LDR camera is able to capture background or the people, not both. The proposed ghost-free HDR merging is able to capture all details in the scene, even with object motion compensation.

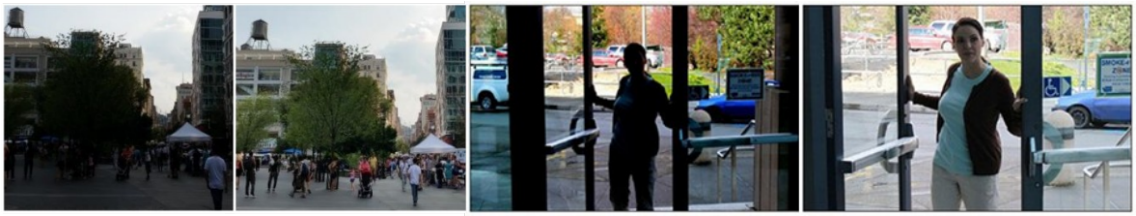


Figure 4.19: An example situation in surveillance applications using LDR cameras. Only the background is well-exposed and the objects of the interest are poorly visible. Left – LDR image, right – HDR image.

Moreover, with the HDR image compression published in Zemcik et al. [70], it is possible to record, display or broadcast the HDR video in real-time, using standard H.264 encoding.

Ghost-free merging in research projects

During the pursuit of my Ph.D., I participated in a number of research projects funded by the Czech Technological Agency (shortly TAČR projects) and the European Union. The TAČR funded projects include V3C, CEPTIS and AITIV projects, where the focus was put on embedded computing platforms for optical inspection in the industry, road traffic surveillance and, in general, to expand the possibilities of video systems for traffic monitoring and traffic detection.

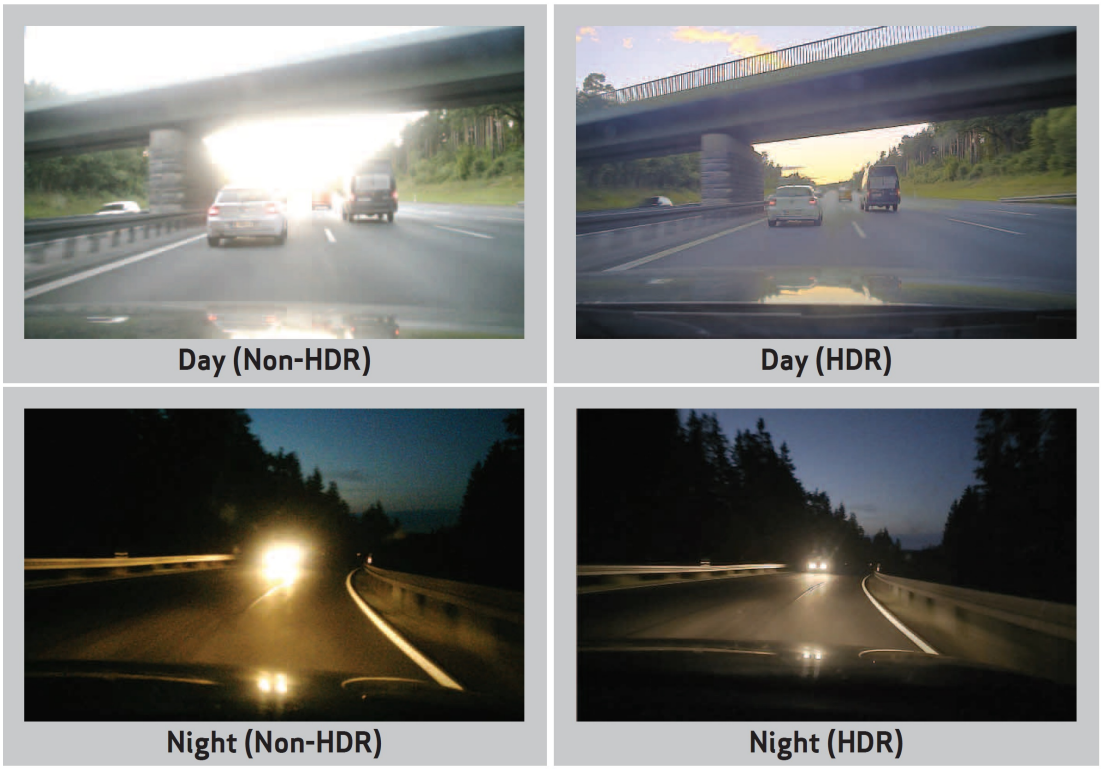


Figure 4.18: Comparison of car camera output with and without HDR mode enabled. Image obtained from *www.ovt.com*.

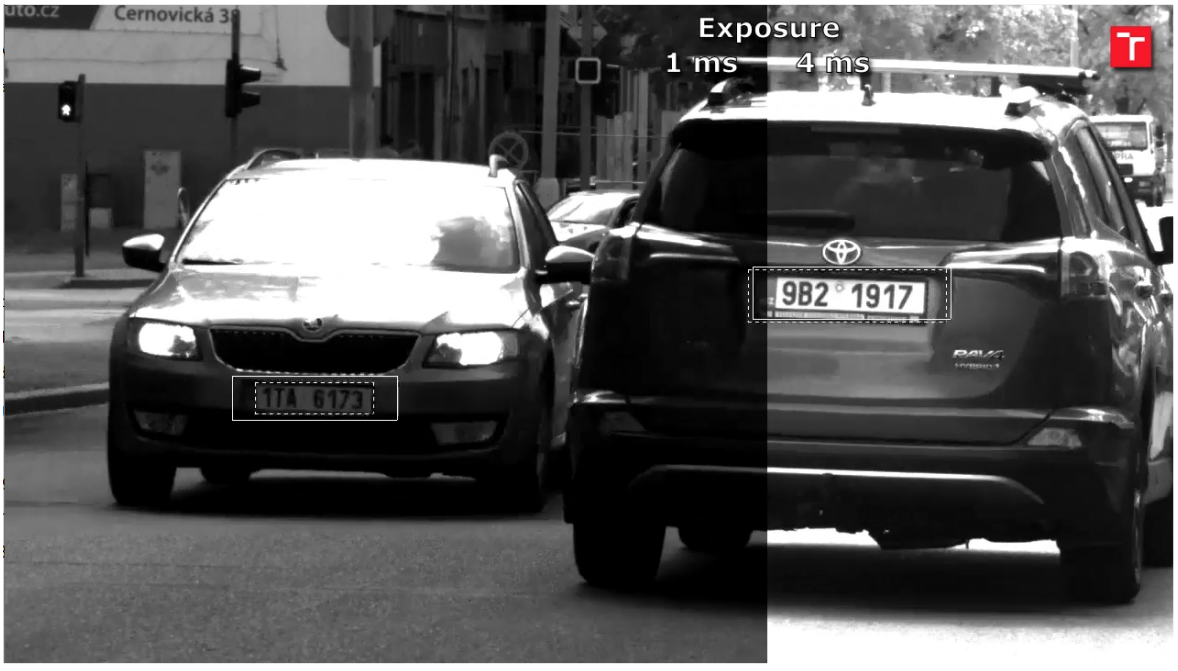


Figure 4.20: A real traffic situation captured by camera prototype by Nosko et al. [42] with embedded licence plate detector [40] (working on LDR images only); Image is obtained from presentation for projects EMC2 and Almarvi. The dash type of rectangle marks, in which source exposition the licence plate was found.

The EU funded projects I participated include CRAFTERS, Almarvi, EMC2 and FitOptiVis projects, which had quite a wide area of focus. The proposed ghost-free acquisition algorithm has been applied within projects Almarvi and FitOptiVis in the applications aimed at surveillance, road traffic monitoring, intelligent cameras and in Industry 4.0. Within this projects, we focused on embedding „intelligence“ into embedded camera systems; it also involved implementation of FPGA object detector [40] into HDR camera(see Figure 4.20), where the „standard“ or ghost-free HDR acquisition significantly improved the ability to detect and detection accuracy comparing to standard LDR mode.

Future work

In the future work, I would like to also continue in this topic and work on the applicability of the proposed solution in practice and in the commercial field, which already started within research projects mentioned in this section. I want to experiment with the current algorithm version and further improve it, for example, extend it for use in non-stationary cameras. It may improve usability, e.g. in surveillance applications, such as automotive and small and inexpensive car cameras.

A different way of development may involve the integration with tone-mapping operators. The information of motion in the form of *Certainty maps* may help to improve tonemapping quality, as the proposed algorithm should identify the motion regions and allow to handle the motion area differently than static part of images. The ghosted areas in the image may produce the adverse effects dependent on the motion of the objects, for example, they may influence the minimum/maximum pixel values or the image histogram and therefore cause flickering within individual tonemapped frames of HDR video.

Chapter 5

Conclusion

In this dissertation work, I focused on the HDR acquisition on embedded devices. The main goal of this thesis was the proof that a multi-exposure ghost-free HDR acquisition algorithm comparable to the state-of-the-art algorithms in quality can be designed for an embedded hardware device and achieves a real-time performance at high resolution. This hypothesis I considered as validated, which was stated in Section 4.5.

I experimentally proved the hypothesis by the successful implementation of proposed ghost-free HDR merging algorithm (Section 4.1) on FPGA based embedded design (Section 4.2). The proposed implementation achieved the expected parameters and is capable of running faster than real-time, up to 96FPS at FullHD resolution (Section 4.4). At the same time, the algorithm produces visual results comparable to the state-of-the-art, as evaluated in the Section 4.3.

The performance evaluation in Section 4.4 shows, that the algorithm performs well even on CPU; single core implementation achieves up to 25.25FPS, which is very fast and multicore CPU could achieve real-time performance as well. Achieved results shows, that even CPU implementation outperformed all the related algorithms.However, essential benefit of this method stand out along with FPGA implementation, which fundamentally affects the power consumption, which is only approx. 1,1% of power comparing to the CPU, as summarized in Section 4.4.1.

The comparison to the state-of-the-art algorithms (Section 4.3) and evaluation of HDR datasets (Section 4.3.1) shows, that proposed algorithm is performing ghost-free HDR merging well and the ghost effect is removed, at the same time have better results and is much more robust than related algorithms. The results are even comparable to the state-of-the-art optical flow-based algorithm, which belongs to the class of performance demanding, offline processing algorithms.

In the future, I would like to continue in this topic and work on the applicability of the proposed solution in practice and in the commercial field, which already started within research projects mentioned in Section 4.6.

Bibliography

[1] An, J., Lee, S. H., Kuk, J. G. and Cho, N. I. A multi-exposure image fusion algorithm without ghost effect. In: *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2011, p. 1565–1568.

[2] Barnes, C., Shechtman, E., Goldman, D. B. and Finkelstein, A. The generalized patchmatch correspondence algorithm. In: Springer. *European Conference on Computer Vision*. 2010, p. 29–43.

[3] Bouderbane, M., Dubois, J., Heyrman, B., Lapray, P.-J. and Ginhac, D. Ghost removing for HDR real-time video stream generation. In: Kehtarnavaz, N. and Carlsohn, M. F., ed. *Real-Time Image and Video Processing 2016*. SPIE, 2016, vol. 9897, p. 112 – 117. DOI: 10.1117/12.2230313. Available at: https://doi.org/10.1117/12.2230313.

[4] Bouderbane, M., Lapray, P.-J., Dubois, J., Heyrman, B. and Ginhac, D. Real-time ghost free HDR video stream generation using weight adaptation based method. In:. September 2016, p. 116–120. DOI: 10.1145/2967413.2967439.

[5] Debevec, P. E. and Malik, J. Recovering High Dynamic Range Radiance Maps from Photographs. In: *ACM Trans. Graph.* 1997. SIGGRAPH '97.

[6] Drago, F., Myszkowski, K., Annen, T. and Chiba, N. Adaptive Logarithmic Mapping For Displaying High Contrast Scenes. *Computer Graphics Forum*. Blackwell Publishing, Inc. 2003, vol. 22, no. 3, p. 419–426. ISSN 1467-8659.

[7] Duan, J., Qiu, G. and Chen, M. Comprehensive Fast Tone Mapping for High Dynamic Range Image Visualization. In: *Pacific Graphics*. 2005.

[8] Duan, J., Bressan, M., Dance, C. and Qiu, G. Tone-mapping High Dynamic Range Images by Novel Histogram Adjustment. *Pattern Recogn.* New York, NY, USA: Elsevier Science Inc. may 2010, vol. 43, no. 5, p. 1847–1862. DOI: 10.1016/j.patcog.2009.12.006. ISSN 0031-3203. Available at: http://dx.doi.org/10.1016/j.patcog.2009.12.006.

[9] Durand, F. and Dorsey, J. Fast Bilateral Filtering for the Display of High-Dynamic-Range Images. In: *ACM Trans. Graph.* ACM, 2002. SIGGRAPH '02.

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

[11] Fattal, R., Lischinski, D. and Werman, M. Gradient Domain High Dynamic Range Compression. *ACM Trans. Graph.* New York, NY, USA: ACM. july 2002, vol. 21, no. 3, p. 249–256. ISSN 0730-0301.

[12] Ferradans, S., Bertalmío, M., Provenzi, E. and Caselles, V. Generation of HDR Images in Non-static Conditions based on Gradient Fusion. In: *VISAPP (1)*. 2012, p. 31–37.

[13] Froehlich, J., Grandinetti, S., Eberhardt, B., Walter, S., Schilling, A. et al. Creating Cinematic Wide Gamut HDR-Video for the Evaluation of Tone Mapping Operators and HDR-Displays. In:. 2014.

[14] Gallo, O., Gelfandz, N., Chen, W.-C., Tico, M. and Pulli, K. Artifact-free High Dynamic Range imaging. In: *2009 IEEE International Conference on Computational Photography (ICCP)*. April 2009, p. 1–7. DOI: 10.1109/ICCPHOT.2009.5559003.

[15] Grosch, T. Fast and robust high dynamic range image generation with camera and object movement. 2006.

[16] Grossberg, M. D. and Nayar, S. K. Determining the camera response from images: what is knowable? *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Nov 2003, vol. 25, no. 11, p. 1455–1467. DOI: 10.1109/TPAMI.2003.1240119. ISSN 0162-8828.

[17] Hu, J., Gallo, O., Pulli, K. and Sun, X. HDR Deghosting: How to Deal with Saturation? In: *2013 IEEE Conference on Computer Vision and Pattern Recognition*. 2013.

[18] Jacobs, K., Loscos, C. and Ward, G. Automatic High-Dynamic Range Image Generation for Dynamic Scenes. *IEEE Computer Graphics and Applications*. March 2008, vol. 28, no. 2, p. 84–93. DOI: 10.1109/MCG.2008.23. ISSN 0272-1716.

[19] Jinno, T. and Okuda, M. Multiple exposure fusion for high dynamic range image acquisition. *IEEE Transactions on Image Processing*. IEEE. 2011, vol. 21, no. 1, p. 358–365.

[20] Kalantari, N. K. and Ramamoorthi, R. Deep High Dynamic Range Imaging of Dynamic Scenes. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2017)*. 2017, vol. 36, no. 4.

[21] Kang, S. B., Uyttendaele, M., Winder, S. and Szeliski, R. High Dynamic Range Video. *ACM Trans. Graph.* New York, NY, USA: ACM. july 2003, vol. 22, no. 3, p. 319–325. ISSN 0730-0301.

[22] Kao, W.-C., Hsu, C.-C., Chen, L.-Y., Kao, C.-C. and Chen, S.-H. Integrating image fusion and motion stabilization for capturing still images in high dynamic range scenes. *IEEE Transactions on Consumer Electronics*. Aug 2006, vol. 52, no. 3, p. 735–741. DOI: 10.1109/TCE.2006.1706464. ISSN 0098-3063.

[23] Karaduzovic Hadziabdic, K., Hasic, T. J. and Mantiuk, R. K. Multi-exposure image stacks for testing HDR deghosting methods. 2017.

[24] Khan, E. A., Akyuz, A. O. and Reinhard, E. Ghost removal in high dynamic range images. In: IEEE. *2006 International Conference on Image Processing*. 2006, p. 2005–2008.

[25] Lapray, P. J., Heyrman, B., Rossé, M. and Ginhac, D. HDR-ARtiSt: High dynamic range advanced real-time imaging system. In: *2012 IEEE International Symposium on Circuits and Systems*. May 2012, p. 1428–1431. DOI: 10.1109/ISCAS.2012.6271513. ISSN 0271-4302.

[26] Lapray, P.-J., Heyrman, B. and Ginhac, D. HDR-ARtiSt: A 1280x1024-pixel Adaptive Real-time Smart camera for High Dynamic Range video. In: *SPIE Photonics Europe*. Brussels, Belgium: [b.n.], April 2014. Available at: https://hal-univ-bourgogne.archives-ouvertes.fr/hal-01196636.

[27] Lapray, P.-J., Heyrman, B. and Ginhac, D. HDR-ARtiSt: an adaptive real-time smart camera for high dynamic range imaging. *J. Real-Time Image Process.* 2016, vol. 12, no. 4, p. 747–762. ISSN 1861-8219.

[28] Liu, C. et al. *Beyond pixels: exploring new representations and applications for motion analysis*. 2009. Dissertation. Massachusetts Institute of Technology.

[29] Mandel, L. Fluctuations of photon beams: the distribution of the photo-electrons. *Proceedings of the Physical Society*. IOP Publishing. 1959, vol. 74, no. 3, p. 233.

[30] Mann, S., Lo, R. C. H., Ovtcharov, K., Gu, S., Dai, D. et al. Realtime HDR (High Dynamic Range) video for eyetap wearable computers, FPGA-based seeing aids, and glasseyes (EyeTaps). In: *2012 25th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*. April 2012, p. 1–6. DOI: 10.1109/CCECE.2012.6335012. ISSN 0840-7789.

[31] Mantiuk, R., Daly, S. and Kerofsky, L. Display Adaptive Tone Mapping. *ACM Trans. Graph.* New York, NY, USA: ACM. august 2008, vol. 27, no. 3, p. 68:1–68:10. ISSN 0730-0301.

[32] Mantiuk, R., Efremov, A., Myszkowski, K. and Seidel, H.-P. Backward compatible high dynamic range MPEG video compression. In: *ACM SIGGRAPH 2006 Papers*. 2006, p. 713–723.

[33] Mantiuk, R., Kim, K. J., Rempel, A. G. and Heidrich, W. HDR-VDP-2: A Calibrated Visual Metric for Visibility and Quality Predictions in All Luminance Conditions. *ACM Trans. Graph.* New York, NY, USA: ACM. july 2011, vol. 30, no. 4, p. 40:1–40:14. DOI: 10.1145/2010324.1964935. ISSN 0730-0301. Available at: http://doi.acm.org/10.1145/2010324.1964935.

[34] Mertens, T., Kautz, J. and Reeth, F. V. Exposure Fusion. In: *Computer Graphics and Applications, 2007. PG '07. 15th Pacific Conference on*. Oct 2007, p. 382–390. ISSN 1550-4085.

[35] Min, T.-H., Park, R.-H. and Chang, S. Histogram based ghost removal in high dynamic range images. In: IEEE. *Multimedia and Expo, 2009. ICME 2009. IEEE International Conference on*. 2009.

[36] Min, T.-H., Park, R.-H. and Chang, S. Noise reduction in high dynamic range images. *Signal, Image and Video Processing*. 2011, vol. 5, no. 3. DOI: 10.1007/s11760-010-0203-7. ISSN 1863-1711. Available at: https://doi.org/10.1007/s11760-010-0203-7.

[37] Mitsunaga, T. and Nayar, S. K. Radiometric self calibration. In: *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*. 1999, vol. 1, p. 380 Vol. 1. ISSN 1063-6919.

[38] Moon, Y., Yong-Min Tai, Cha, J. H. and Lee, S. A simple ghost-free exposure fusion for embedded HDR imaging. In: *2012 IEEE International Conference on Consumer Electronics (ICCE)*. 2012, p. 9–10.

[39] Musil, M., Nosko, S. and Zemcik, P. De-ghosted HDR video acquisition for embedded systems. *Journal of Real-Time Image Processing*. Springer. 2020, p. 1–10.

[40] Musil, P., Juránek, R., Musil, M. and Zemčík, P. Cascaded Stripe Memory Engines for Multi-Scale Object Detection in FPGA. *IEEE Transactions on Circuits and Systems for Video Technology*. 2020, vol. 30, no. 1, p. 267–280.

[41] Nosko, S., Musil, M., Musil, P. and Zemcik, P. True HDR Camera with Bilateral Filter Based Tone Mapping. In: *Proceedings of the 33rd Spring Conference on Computer Graphics*. New York, NY, USA: ACM, 2017, p. 15:1–15:9. SCCG '17. DOI: 10.1145/3154353.3154367. ISBN 978-1-4503-5107-2. Available at: http://doi.acm.org/10.1145/3154353.3154367.

[42] Nosko, S., Musil, M., Zemcik, P. and Juranek, R. Color HDR video processing architecture for smart camera. *Journal of Real-Time Image Processing*. Jul 2018. DOI: 10.1007/s11554-018-0810-z. ISSN 1861-8219. Available at: https://doi.org/10.1007/s11554-018-0810-z.

[43] Orozco, R. R., Martin, I., Loscos, C. and Vasquez, P.-P. Full high-dynamic range images for dynamic scenes. In: International Society for Optics and Photonics. *Optics, Photonics, and Digital Technologies for Multimedia Applications II*. 2012, vol. 8436, p. 843609.

[44] Pece, F. and Kautz, J. Bitmap movement detection: HDR for dynamic scenes. In: IEEE. *Visual Media Production, 2010 Conference on*. 2010, p. 1–8.

[45] Popadić, I., Todorović, B. M. and Reljin, I. Method for HDR-like imaging using industrial digital cameras. *Multimedia Tools and Applications*. May 2017, vol. 76, no. 10, p. 12801–12817. DOI: 10.1007/s11042-016-3692-8. ISSN 1573-7721. Available at: https://doi.org/10.1007/s11042-016-3692-8.

[46] Raman, S., Kumar, V. and Chaudhuri, S. Blind De-ghosting for Automatic Multi-exposure Compositing. In: *ACM SIGGRAPH ASIA 2009 Posters*. USA: ACM, 2009. SIGGRAPH ASIA '09.

[47] Reinhard, E., Stark, M., Shirley, P. and Ferwerda, J. Photographic Tone Reproduction for Digital Images. *ACM Trans. Graph.* New York, NY, USA: ACM. july 2002, vol. 21, no. 3, p. 267–276. ISSN 0730-0301.

[48] Robertson, M. A., Borman, S. and Stevenson, R. L. Estimation-theoretic approach to dynamic range enhancement using multiple exposures. *J. Electronic Imaging*. 2003, vol. 12, no. 2, p. 219–228.

[49] Sakakibara, M., Kawahito, S., Handoko, D., Nakamura, N., Satoh, H. et al. A high-sensitivity CMOS image sensor with gain-adaptive column amplifiers. *J. of Solid-State Circuits*. May 2005, vol. 40, no. 5. ISSN 0018-9200.

[50] Sen, P., Kalantari, N. K., Yaesoubi, M., Darabi, S., Goldman, D. B. et al. Robust Patch-Based HDR Reconstruction of Dynamic Scenes. *ACM Transactions on Graphics (TOG) (Proceedings of SIGGRAPH Asia 2012)*. 2012, vol. 31, no. 6, p. 203:1–203:11.

[51] Sidibe, D., Puech, W. and Strauss, O. Ghost detection and removal in High Dynamic Range Images. In: *2009 17th European Signal Processing Conference*. 2009, p. 2240–2244.

[52] Silk, S. and Lang, J. Fast High Dynamic Range Image Deghosting for Arbitrary Scene Motion. In: *Proceedings of Graphics Interface 2012*. Toronto, Ont., Canada, Canada: Canadian Information Processing Society, 2012, p. 85–92. GI '12. ISBN 978-1-4503-1420-6. Available at: http://dl.acm.org/citation.cfm?id=2305276.2305291.

[53] Spaulding, K., Woolfe, G. and Joshi, R. Using a Residual Image to Extend the Color Gamut and Dynamic Range of an sRGB Image. In:. August 2003, vol. 28, p. 307–314. DOI: 10.1002/col.10160.

[54] Srikantha, A. and Sidibé, D. Ghost detection and removal for high dynamic range images: Recent advances. *Signal Processing: Image Communication*. 2012, vol. 27, no. 6, p. 650 – 662. DOI: https://doi.org/10.1016/j.image.2012.02.001. ISSN 0923-5965. Available at: http://www.sciencedirect.com/science/article/pii/S0923596512000306.

[55] Tamburrino, D., Alleysson, D., Meylan, L. and Süsstrunk, S. Digital camera workflow for high dynamic range images using a model of retinal processing. In: *IST/SPIE Electronic Imaging: Digital Photography IV*. 2008, vol. 6817, LCAV-CONF-2007-030.

[56] Tang, X., Qian, Y., Kong, X. and Wang, H. A high-dynamic range CMOS camera based on dual-gain channels. *Journal of Real-Time Image Processing*. may 2019. DOI: 10.1007/s11554-019-00877-8.

[57] Tao Ai, Ali, M. A., Steffan, G., Ovtcharov, K., Zulfiqar, S. et al. Real-time HDR video imaging on FPGA with compressed comparametric lookup tables. In: *2014 IEEE 27th Canadian Conference on Electrical and Computer Engineering (CCECE)*. 2014, p. 1–6.

[58] Tocci, M. D., Kiser, C., Tocci, N. and Sen, P. A Versatile HDR Video Production System. In: *ACM SIGGRAPH 2011 Papers*. USA: [b.n.], 2011. SIGGRAPH '11. ISBN 978-1-4503-0943-1.

[59] Tursun, O. T., Akyüz, A. O., Erdem, A. and Erdem, E. The State of the Art in HDR Deghosting: A Survey and Evaluation. *Computer Graphics Forum*. 2015, vol. 34, no. 2. DOI: 10.1111/cgf.12593. ISSN 1467-8659.

[60] Tursun, O. T., Akyüz, A. O., Erdem, A. and Erdem, E. An Objective Deghosting Quality Metric for HDR Images. *Comput. Graph. Forum*. Chichester, UK: The Eurographs Association & John Wiley & Sons, Ltd. may 2016, vol. 35, no. 2, p. 139–152. DOI: 10.1111/cgf.12818. ISSN 0167-7055. Available at: https://doi.org/10.1111/cgf.12818.

[61] Ureña, R., Martínez Cañada, P., Gómez López, J. M., Morillas, C. and Pelayo, F. Real-time tone mapping on GPU and FPGA. *EURASIP Journal on Image and Video Processing*. 2012, vol. 2012, no. 1, p. 1. ISSN 1687-5281.

[62] Vytla, L., Hassan, F. and Carletta, J. E. A Real-time Implementation of Gradient Domain High Dynamic Range Compression Using a Local Poisson Solver. *J. Real-Time Image Process.* Secaucus, NJ, USA: Springer-Verlag New York, Inc. june 2013, vol. 8, no. 2, p. 153–167. ISSN 1861-8200.

[63] Wang, C. and Tu, C. An exposure fusion approach without ghost for dynamic scenes. In: *2013 6th International Congress on Image and Signal Processing (CISP)*. Dec 2013, vol. 2, p. 904–909. DOI: 10.1109/CISP.2013.6745293.

[64] Ward, G. Fast, Robust Image Registration for Compositing High Dynamic Range Photographs from Handheld Exposures. *JOURNAL OF GRAPHICS TOOLS*. 2003, vol. 8, p. 17–30.

[65] Ward, G. and Simmons, M. Subband encoding of high dynamic range imagery. In:. August 2004, p. 83–90. DOI: 10.1145/1185657.1185684.

[66] Wu, S., Xie, S., Rahardja, S. and Li, Z. A robust and fast anti-ghosting algorithm for high dynamic range imaging. In: *2010 IEEE International Conference on Image Processing*. Sept 2010, p. 397–400. DOI: 10.1109/ICIP.2010.5654196. ISSN 1522-4880.

[67] Wu, S., Xu, J., Tai, Y.-W. and Tang, C.-K. Deep high dynamic range imaging with large foreground motions. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, p. 117–132.

[68] Xie, D. and Wang, Y. High definition wide dynamic video surveillance system based on FPGA. In: *2017 IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*. 2017, p. 2403–2407.

[69] Yan, Q., Zhang, L., Liu, Y., Zhu, Y., Sun, J. et al. Deep HDR Imaging via A Non-Local Network. *IEEE Transactions on Image Processing*. 2020, vol. 29, p. 4308–4322.

[70] Zemcik, P., Musil, P. and Musil, M. Real-time HDR video processing and compression using an FPGA. *High Dynamic Range Video: Concepts, Technologies and Applications*. 2016, p. 145–154.

[71] Zhang, W. and Cham, W.-K. Gradient-directed composition of multi-exposure images. In: IEEE. *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2010, p. 530–536.

[72] Zhang, Y., Agrafiotis, D. and Bull, D. R. High Dynamic Range image video compression a review. In: *2013 18th International Conference on Digital Signal Processing (DSP)*. July 2013, p. 1–7. DOI: 10.1109/ICDSP.2013.6622714. ISSN 1546-1874.

[73] Zhao, H., Shi, B., Fernandez Cull, C., Yeung, S.-K. and Raskar, R. Unbounded High Dynamic Range Photography using a Modulo Camera. In: *ICCP*. 2015.

[74] Zicong Mai, Mansour, H., Mantiuk, R., Nasiopoulos, P., Ward, R. et al. Optimizing a Tone Curve for Backward-Compatible High Dynamic Range Image and Video Compression. *IEEE Transactions on Image Processing*. june 2011, vol. 20, no. 6, p. 1558–1571. DOI: 10.1109/TIP.2010.2095866.

[75] Zimmer, H., Bruhn, A. and Weickert, J. Freehand HDR Imaging of Moving Scenes with Simultaneous Resolution Enhancement. *Computer Graphics Forum (Proceedings of Eurographics)*. Blackwell Publishing Ltd. 2011, vol. 30, no. 2, p. 405–414. Available at: http://www.mia.uni-saarland.de/Research/SR-HDR/.

[76] Zitová, B. and Flusser, J. Image registration methods: a survey. *Image and Vision Computing*. 2003, vol. 21, no. 11, p. 977 – 1000. DOI: https://doi.org/10.1016/S0262-8856(03)00137-9. ISSN 0262-8856. Available at: http://www.sciencedirect.com/science/article/pii/S0262885603001379.