Lens Flare Attenuation Accelerator Design with Deep Learning and High-Level Synthesis

1st David Fosca Gamarra
Department of Electronic Systems
NTNU
Trondheim, Norway

2nd Per Gunnar Kjeldsberg
Department of Electronic Systems
NTNU
Trondheim, Norway

3rd Henrik Sundbeck
Sony Semiconductors EU
Oslo, Norway

***Abstract*—Lens flare artifacts are undesired visual distortions caused by stray light, which can negatively impact the integrity and quality of an image. These artifacts pose a significant challenge in industrial applications like automotive and surveillance, where the quality and reliability of input images from cameras are crucial. Artificial intelligence, particularly deep learning neural networks, have shown promising results in attenuating lens flare. In this work, a synthetic flare dataset is generated, and an iterative training process that includes evaluation of transfer learning is employed to develop FlareNet, the first compact and lightweight U-Net based model for lens flare reduction. The FlareNet architecture, with less than 150,000 parameters comprising convolutional layers, demonstrates improvement in image quality by reducing flare artifacts on synthetic test images and real-life images, indicating its potential for achieving visually satisfactory results despite having less than 0.5% of the weights of the state-of-the-art neural architecture used for this same application. To demonstrate the viability of using a model such as FlareNet as a hardware accelerator, the neural network is implemented in C++ using Vitis HLS. Synthesis and validation are performed using the Vitis tool, and reports are analyzed while experimenting with HLS optimization directives. Resource utilization of less than 20% on a Zeus Zynq UltraScale FPGA is shown but further work is needed to optimize the design for real-time applications and effectively deploy the solution on an FPGA.**

***Index Terms*—Lens flare artifacts, Deep Learning, High Level Synthesis (HLS).**

I. Introduction

Photographs of scenes with a strong light source within or near to the optics system's field of view tend to present lens flare artifacts. Although lens flare is just one type of stray light phenomenon, it is particularly problematic as it obstructs content in the image, reduces contrast and color diversity, and negatively impacts overall image quality and interpretability [1]. It can affect the performance of systems in various domains, such as healthcare, industrial, security, and automotive. There are approaches to physically modify the optical system to prevent unwanted light from entering through the lenses. For instance, this can be achieved through the use of baffle lenses, which block stray light from entering the optical system. However, this approach is typically costly compared to a post-processing technique. Stray light is an unwanted electromagnetic radiation which effect on an image can be modeled mathematically by describing the observed image as the convolution between a underlying clean image and a filter, also known as the Point Spread Function (PSF) [2]. The PSF is the impulse response of an optical system, characterizing the system's response to an individual point of light source. Therefore, it is theoretically possible to retrieve the underlying image by means of applying the inverse operation of convolution, also known as deconvolution. However, the reconstruction deconvolution algorithm can only be applied to images taken by the same camera system used to characterize the PSF [2]. In addition, deconvolution algorithms for stray light attenuation have primarily been developed and tested for static camera system setups, such as in microscopy, astronomy, or healthcare applications [3]. Deploying these solutions in dynamic and rapidly changing environments, such as in industrial applications, introduces additional challenges. Factors like camera motion, changing lighting conditions, and lens wear and tear can significantly affect the PSF, necessitating frequent PSF calibration for each of the RGB channels [4]. Iterative deconvolution algorithms, like Richardson-Lucy, have shown promise for image reconstruction in the presence of varying PSFs. However, determining the optimal number of iterations for achieving the best reconstruction is not straightforward. If the stopping criterion is not chosen carefully, these algorithms may introduce artifacts in the restored image, compromising the quality of the reconstruction [5].

On the other hand, deep learning approaches, such as convolutional neural networks (CNNs), have shown remarkable results in image restoration for lens flare artifacts attenuation in recent years [6] [7] [8]. Through deep learning there is no need for PSF characterization of a specific camera system, enabling a camera agnostic solution for flare attenuation. Previous studies have demonstrated promising outcomes in flare attenuation tasks using well-established convolutional neural network (CNN) architectures, such as U-Net [9]. Additionally, they successfully demonstrated the feasibility of using synthetic flare datasets to train deep learning models for accurate predictions on real-life data. By leveraging synthetic data during the training phase, the deep learning model can learn and capture the essential features and characteristics of flare artifacts. However, implementing deep learning architectures like convolutional networks for image processing inference in edge devices also poses challenges. Traditional software-based implementations of image processing algorithms are not optimal for real-time systems [10]. Image processing via

979-8-3503-3757-0/23/$31.00 2023 IEEE, Author's postpring for university database

hardware implementation poses a more viable solution for improving performance of image processing systems through the development of specialized hardware accelerators. Field-Programmable Gate Arrays (FPGAs) are integrated circuits that allows for the development of custom logic for rapid prototyping of a digital design solution. GPUs (Graphics Processing Units) are another interesting option for image acceleration, as they provide high throughput, but they are not well-suited for low-power applications. In this paper we present a proof-of-concept design that could potentially be used in an FPGA based accelerator using High Level Synthesis (HLS) and deep learning.

II. Theoretical Background

A. Deep Learning for Flare Attenuation

Deep learning has achieved remarkable success in various domains. However, applying it to flare attenuation tasks has been limited by the scarcity of real datasets available for training such models. Acquiring a large number of aligned images with and without flare from real scenarios is impractical due to the need for consistent photographic conditions (lighting, camera angle, exposure time, and other factors) between images that cannot be controlled in real-world scenarios. To overcome this challenge, several research works, such as [6], [7], and [8], have addressed the problem by leveraging high-quality synthetic datasets that incorporate a wide diversity of flare images merged with ground-truth (GT) clean scenes.

To validate the use of synthetic datasets, all of these studies trained deep neural networks and evaluated the performance of different architectures. For instance, the work done in [6] found that the U-Net architecture produced the best results for flare attenuation. During training, they employed two types of loss calculations: image loss and flare loss using Mean Squared Error (MSE) and Mean Absolute Error (MAE) metrics respectively. The former encourages the predicted image to closely resemble the ground-truth, while the latter encourages the avoidance of introducing artifacts in the predictions. The training process involved approximately 60 epochs on a dataset of 20,000 samples, using the Adam optimizer with a fixed learning rate of 0.0001. To measure the quality of the reconstructed images, Structural Similarity Index (SSIM) was used as a metric, comparing the predictions to a baseline defined by the input image (with flare) and ground-truth (without flare) from the synthetic dataset. They achieved a Structural Similarity Index (SSIM) of 0.994, surpassing the baseline SSIM of 0.843, indicating that the predicted image closely resembles the input image, mitigating the flare artifacts. In addition, they also conducted an evaluation on a real dataset visually showing the effectiveness of the model at attenuating lens flare.

The existing deep learning flare attenuation approaches have not been designed for deployment in constrained embedded systems, however, due to their high complexity, assessing the performance of the solution only on CPU.

B. Compact U-Net architectures

U-Net is a state-of-the-art deep learning auto-encoder architecture used for image segmentation [9]. An auto-encoder is a special type of neural network that copies its input value to the output by learning how to compress the input data while minimizing reconstruction error. Originally developed for biomedical image segmentation, U-Net gained popularity in different domains due to its combination of speed and precision.

Researchers have made efforts to reduce the complexity of U-Net models. For instance, the Squeeze U-Net model proposed in [11] achieves similar accuracy to the original U-Net while having only 2.59 million trainable parameters compared to the original model [9] with 30 million. Another approach to reducing complexity is presented in [12], where the authors propose two U-Net-like architectures called C-UNet and C-UNet++. These models reduce complexity by removing convolutional stages compared to the original architecture. Additionally, they utilize separable depthwise convolutional layers, which significantly reduce the number of parameters while maintaining the filtering capability. The depth of the filters is also reduced, with the largest depth being 32, compared to the original U-Net's layers with 128, 256, 512, and 1024 depth filters. The authors find a good balance between size and accuracy, with the C-UNet model having 51,113 parameters and the C-UNet++ model having as few as 9,129 parameters with only 4 types of convolutional layers (2D convolution, 2D depthwise separable convolution, 2D transpose convolution and 2D max-pooling), which can be seen in Figure 1.

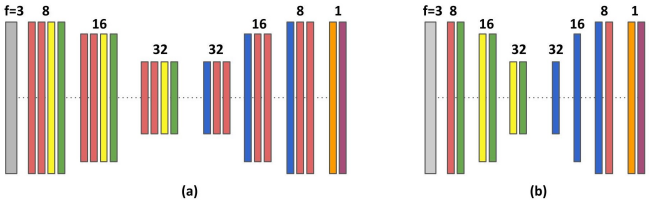

Fig. 1. Compact U-Net based architectures proposed by [12]: (a) C-UNet and (b) C-UNet++. red: conv3x3 + ReLU, yellow: depthwise separable conv3x3 + ReLU, green: 2x2 max pool, orange: conv1x1 sigmoid, blue: 2x2 transpose convolution, purple: output

C. High-Level Synthesis (HLS)

While FPGAs offer numerous advantages for developing tailored accelerators, it is important to acknowledge that implementing FPGA solutions requires highly specialized expertise in hardware description languages such as VHDL and Verilog. Particularly for complex algorithms like image or signal processing, the overall design efforts and time to market can significantly increase due to the inherent challenges in designing and validating RTL (Register Transfer Level) designs. However, platforms such as those provided by XILINX can greatly assist developers in this journey. One notable tool is Vitis HLS. HLS is a powerful technology that can efficiently

transform behavioral logic described in high-level languages, such as C/C++ or SystemC, into digital hardware. It offers an easier and quicker way of exploring design options, resulting in more effective hardware solutions with shorter time-to-market [13]. The result of the optimization process depends on user directives and constraints such as latency, hardware resource utilization, and throughput, as well as the available microelectronic technology.

III. Design and Implementation

The primary objective of this work is to evaluate potential image post-processing solutions for flare attenuation. However, several considerations need to be taken into account during this process to assess the prototype design and its results, with a focus on identifying areas for improvement and future optimizations.

A. Design Considerations

- Real-time performance: The solution should aim to achieve a processing speed close to 30 frames per second (FPS) to be suitable for real-time applications.
- Camera-agnostic solution: The solution should not only be designed for one type of camera or optic system.
- Embedded system compatibility: The solution should be designed to be deployable on an embedded system with constrained resources. It should be optimized to operate efficiently within these limitations.
- Subtle flare attenuation: The solution should be capable of attenuating flare artifacts in a subtle manner, ensuring that no additional artifacts or distortions are introduced into the image during the process.
- Flare artifact type: The solution should be designed to attenuate flare artifacts caused by the presence of a light source within or in close proximity to the camera's field of view.

B. Dataset

In this work, the flare dataset is selected from [6] because of its high quality, diversity of lens flare patterns and the fact that it has already been used for training a deep learning model. The other public flare dataset [8] is especially created for night applications and does not include real flares. Therefore, the selected dataset comprises 2000 synthetic flares and 3000 real flares captured by a camera rotating around a light source in a dark room. Moreover, the scene dataset is based on the Flickr 30k [15]. Figure 2 shows samples of both datasets.

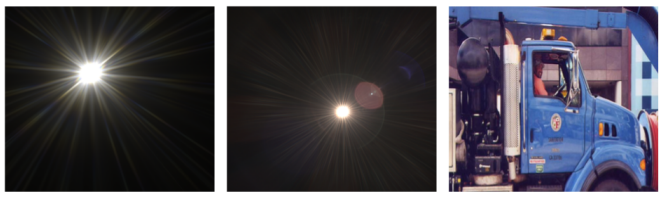

Fig. 2. Sample images from the flare (synthetic and real) and the scene datasets.

The procedure to generate the synthetic dataset involves several steps based on a method proposed in [6]. First, a scene and a flare image are randomly selected from their respective datasets. Then, the flare image undergoes random scaling, rotation, and color augmentation procedures. Finally, the modified flare image is overlaid onto the scene to create a composite image, which is used as the input for the deep learning model. On the other hand, the corresponding original image of the scene is used to create the ground-truth image. This process is repeated multiple times with different random combinations of flare and scene images to create a diverse and complex synthetic dataset of 32,000 pairs of images (with and without flare).

The neural network should not focus on removing the light source, otherwise it will be a waste of model capacity and the resulting image would be susceptible to artifacts appearing where the light source (saturated pixels) is supposed to be as explained in [6]. Therefore, a new ground-truth is computed by using the flare and its corresponding scene, where the pixels with an RGB value above a defined saturation threshold (0.97) are passed into the ground-truth scene. The result of this process is illustrated in Figure 3.

Fig. 3. Example of scene merged with flare (left) and new ground-truth with saturated pixels (right).

C. Model Definition and Training

The evaluation started with a smaller model inspired by the structure of U-Net and implemented using transfer learning. In this case, MobileNetv2 [16], specifically designed for applications with limited computing resources, was selected as the backbone, and then retrained on the flare dataset. To further reduce the complexity of the transfer learning network, the scaling parameter (alpha) of MobileNetv2 was manually adjusted to reduce the neural network complexity. The alpha parameter controls the number of channels in the network, with a value of 1.0 indicating maximum width. In this work, an alpha value of 0.35 was used, significantly reducing the original design network from around 2 million parameters to 141,646 parameters.

Although the transfer learning-based network, referred to as FlareNet-TL, demonstrates promising results, part of its architecture inherited from MobileNetv2 poses an extra complexity when it comes to hardware implementation due to its higher number of layers and parameters. To address this limitation and seek a simpler yet effective neural network, a new model without transfer learning is proposed inspired on more compact models such as the C-UNet.

Through several iterations, different hyper-parameters and CNN layers were evaluated to identify an optimized architecture with good performance while keeping the parameter count limited following a similar approach to [12]. The best architecture, depicted in Figure 4, emerged from this iterative process. During the exploratory phase, significant modifications were made to the network's depth, filter sizes, and inclusion of skip connections. The resulting model, referred to as FlareNet-simple, has a total of 92,051 parameters.

D. Hardware Implementation

As part of the-proof-of-concept, the FlareNet-simple architecture is implemented using HLS to estimate the hardware resources required to deploy it as a digital circuit on an FPGA. The overall data flow of the neural network model is shown in Figure 4. The architecture consists of five different types of layers: i) 2D Convolution, ii) 2D Max-Pooling, iii) 2D Depth-wise separable convolution, iv) 2D Transpose Convolution, and v) Adding. Additionally, each layer considers buffers to cache input and intermediary values during the inference process. The data is transmitted from layer to layer using stream data types and each layer is implemented as templated functions to be called with corresponding values for parameters such as input/output size, input/output depth, kernel weights, among others.

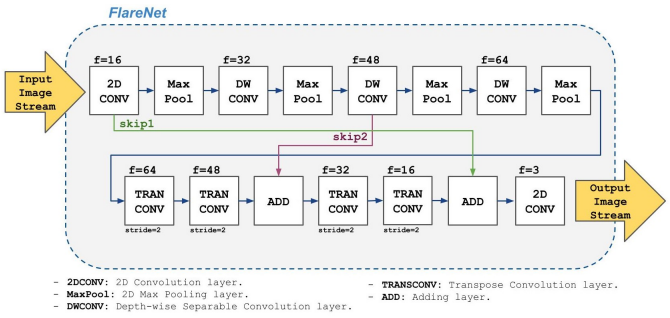

Fig. 4. Diagram of the structure of the FlareNet model to be implemented in HLS.

Selecting the appropriate data type representation for the input and output image values as well as the model weights is very important as it will impact the use of resources and the latency of the design. In the context of this application, we have allocated 10 bits for the integer part and 8 bits for the decimal places. The selection of the number of bits is done based on comparing the image inference results from the implementation in C++ with the inference results directly from the deep learning framework with full bit representation (float32) to select the smallest fixed-point representation that generates minimum differences between them. This is essential to ensure that values in internal layers during inference calculation have sufficient bit range to prevent truncation or overflow issues as it is possible to see in Figure 5 if less bits are used for the integer section.

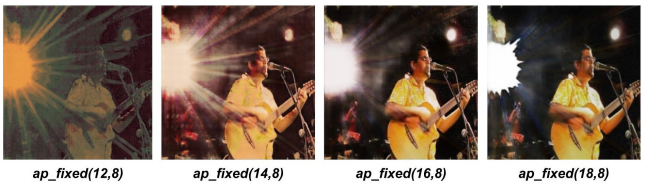

Fig. 5. Examples of different FlareNet inference results depending on the fixed-point resolution.

E. Inference in GPU

The performance of the trained model on an AI accelerator, is evaluated using two GPUs with built in Deep Learning support from NVIDIA: RTX3060 and Jetson Nano GPU. To run inference on both devices, an application is built using C++, OpenCV, and ONNX-runtime. ONNX is an open-source ecosystem that allows representing machine learning models with a standardized set of operators across multiple frameworks, providing access to hardware optimizations while maximizing performance. In the case of running the model on NVIDIA GPUs, ONNX utilizes CUDA for optimizations.

IV. Results and Discussion

A. FlareNet-TL vs FlareNet-simple

A test dataset comprising 25,425 synthetic test images is used to evalute the performance of the models. The results presented in Table I demonstrates that both models have the capacity to reduce flare by increasing the similarity between the ground-truth and the inference image as seen by the SSIM metric or decreasing the difference between both of them as proved by the MSE and MAE metrics.

However, it is important to note that the current dataset consists solely of synthetic images. To ensure a comprehensive evaluation, real-world images containing actual flare artifacts should also be considered in the assessment process. As expected, the transfer learning-based model shows superior performance compared to the simpler one.

TABLE I
Inference vs Ground-truth (GT) Accuracy

	Input vs GT	**TL vs GT**	**Simple vs GT**
SSIM	0.803	0.881	0.765
MAE	0.105	0.073	0.087
MSE	0.028	0.011	0.018

B. Model Inference

The FlareNet models are restoring the image in two ways: i) attenuating the flare, and ii) restoring the RGB values of the image. To this end, the network is trained to minimize the difference between the restored image and the original image in terms of structural similarity, as measured by the SSIM function. This helps ensure that the network produces visually pleasing and natural-looking images after flare attenuation.

Real-life cases of lens flare artifacts are used to assess the generalization capability of the FlareNet models. It is

Fig. 6. Left: input image, middle: ground-truth, right: FlareNet-TL prediction.

Fig. 7. Left: input image, middle: ground-truth, right: FlareNet-TL prediction.

important to note that since there is no ground-truth available for comparison, there is no objective metric to quantitatively measure the extent of attenuation achieved by the model. However, a visual inspection of the following images reveals a noticeable reduction in the intensity of the flares, indicating that the model successfully attenuates them to some degree. It is important to note that although the flares in the real images are originating from the sun as the primary light source, this serves as a proof-of-concept. It is crucial to further evaluate the model's performance on different types of flares originating from various light sources. Figures 8 and 9 (taken by Xiaomi, and Iphone camera, respectively) demonstrate the superior performance of FlareNet-TL in attenuating flare while effectively avoiding the generation of additional artifacts in the image. In comparison, FlareNet-simple occasionally produces black spots around the light source. These findings reinforce the results obtained from the synthetic dataset in Section IV-A, indicating that FlareNet-TL exhibits better generalization capabilities when encountering new data.

Fig. 8. Left: input, middle: FlareNet-TL prediction, right: FlareNet-simple prediction.

It is worth mentioning that the current state-of-the-art model for flare attenuation [6] utilizes the original U-Net architecture, as defined in [9], with 23 convolutional layers and nearly 30 million learnable parameters. It is evident that this model has not been designed for deployment on memory constraint devices such as embedded systems. In comparison, the FlareNet-TL model employs roughly less than 0.5% of the parameters

Fig. 9. Left: input, middle: FlareNet-TL prediction, right: FlareNet-simple prediction.

found in the state-of-the-art flare attenuation model.

Furthermore, the images presented below in Figures 10 and 11 provide visual comparisons between the predictions made by FlareNet-TL (middle image) and the state-of-the-art flare attenuation model (right image) from [6]. As expected, due to its higher complexity, the predictions made by the state-of-the-art model are superior, nearly eliminating any signs of flare. However, FlareNet still strives to blend the flare artifacts into the rest of the image and improve color contrast without introducing artifacts, especially for pixels that are further away from the light source.

Fig. 10. Left: input, middle: FlareNet-TL prediction, right: state-of-the-art model [6].

Fig. 11. Left: input, middle: FlareNet-TL prediction, right: state-of-the-art model [6].

C. Hardware Synthesis

This section presents the findings of the hardware synthesis of the FlareNet-simple model using Vitis HLS. The main objectives are to analyze the performance of the synthesized solution and assess the impact of different optimization directives on key metrics such as latency and resource utilization (area). This analysis provides an indication of the required hardware resources as well as performance, based on Vitis HLS simulation and synthesis tool.

During synthesis, certain parameters and constraints are specified. The clock frequency constraint is set to 300 MHz.

Additionally, the target FPGA board is selected for synthesis. The Zeus Zynq UltraScale FPGA board (target device: XQZU11EG-FFRC1760-2-i) is chosen as it is an industrial graded device used in automotive and artificial intelligence applications. These specifications guide the synthesis tool to generate the optimized hardware implementation for the specified FPGA target.

An initial implementation was synthesized disabling all optimization directives, e.g., dataflow, pipeline, loop unrolling, and array partitioning. This resulted in the HLS tool generating a sequential design, where tasks within layers and functions are executed sequentially, one after the other. Although this approach allows for resource re-utilization, leading to lower resource consumption/area (as indicated in Table II), it also increases the latency (as observed in Table III).

TABLE II
LATENCY ANALYSIS - NO OPTIMIZATIONS - FLOATING POINT

	HW Latency (cycles)	HW Latency (ms)
Min	4889194250	16134
Max	5584202570	18428

TABLE III
UTILIZATION ANALYSIS - NO OPTIMIZATIONS - FLOATING POINT

	BRAM 18K	DSP	FF	LUT
Total	31	5	10962	26507
Available	1200	2928	597120	298560
Util. (%)	31	0.002	1	8

Next, the pipeline and dataflow optimization directives are applied together, to allow for a more optimized implementation taking advantage of both parallel execution within loops and concurrent execution of tasks within layers and functions of the neural network architecture. By using both directives together, the HLS tool can generate a design that benefits from both intra-loop parallelism (achieved through the pipeline directive) and inter-task parallelism (achieved through the dataflow directive). The implementation of this approach has notably reduced latency, as demonstrated in Table IV, when contrasted with the latency values provided in Table II. In this comparison, we observe an impressive 36-fold reduction in latency, ultimately enabling our solution to achieve an approximate processing rate of 2 frames per second. However, it also requires additional resources to work concurrently, as evident in Table III compared to Table V. The utilization of Flip-Flops (FF), Digital Signal Processing (DSP), and Look-Up Tables (LUT) shows a significant increase. This increase is attributed to the replication of functional units to enable concurrent usage, unlike the sequential implementation where units could be reused. Additionally, pipelining requires instantiating extra registers, known as pipeline registers, as well as new pipeline control signals.

Due to time constraints, limited attempts were made to further optimize the solution, but it became evident that a different approach was required for the buffer structure and related functions as they seemed to create a memory bottleneck that limits further parallelism done by the HLS tool. By delving deeper into this area, there's potential to achieve even greater reductions in latency.

TABLE IV
LATENCY ANALYSIS - DATAFLOW AND PIPELINE

	HW Latency (cycles)	HW Latency (ms)
Min	155277746	512
Max	155277746	512

TABLE V
UTILIZATION ANALYSIS - DATAFLOW AND PIPELINE

	BRAM 18K	DSP	FF	LUT
Total	199	55	14797	29972
Available	1200	2928	597120	298560
Util. (%)	16	1	2	10

D. Inference in GPU

The model's performance was evaluated on two GPUs, with the objective of comparing the execution time on different hardware accelerators. On a laptop with a medium-end GPU, the model achieved an inference speed of 32 frames-per-second, while on a lower-end GPU like the Jetson Nano, it achieved 6 frames-per-second. Considering that these GPUs operate at frequencies between 650 MHz and 950 MHz, lower inference times would be expected at the cost of higher power consumption with respect to an FPGA implementation.

V. CONCLUSION

The primary objective of this study was to evaluate an approach for reducing lens flare artifacts through deep learning based image post-processing techniques. A synthetic flare dataset was generated, and an iterative training process was employed to develop the first compact and lightweight U-Net based model for lens flare reduction, named FlareNet with and without a transfer learning component. Noteworthy insights were gained during the process, including the benefits of transfer learning, and the selection of an appropriate loss metric. It was found that, for a small model, using the SSIM as a loss metric yielded effective results in reducing flare without introducing additional artifacts in the restored image. Both versions of the FlareNet model (with and without transfer learning) demonstrated improvement in image quality on the testing dataset, with a modest parameter count of less than 150,000 and a simple neural network architecture. Furthermore, as part of the proof-of-concept, the simple FlareNet model version was implemented in C++ using Vitis HLS to profile the required resources and performance when deployed as a digital circuit on an FPGA. Results demonstrated that for the selected FPGA and a clock frequency of 300 MHz, the inference time is approximately 512 ms (equivalent to approximately 2 frames-per-second) with minimal resource utilization well within the device's limits.

GitHub Project Repository:
Flare Attenuation Filter

REFERENCES

[1] J.-O. Park, W.-K. Jang, S.-H. Kim, H.-S. Jang, and S.-H. Lee, "Stray light analysis of high resolution camera for a low-earth-orbit satellite," J. Opt. Soc. Korea, vol. 15, no. 1, pp. 52–55, Mar 2011.

[2] A. Pirinen and A. Toytziaridis, "Stray light compensation in optical systems," Master's Theses in Mathematical Sciences, 2015.

[3] L. Clermont, W. Uhring, and M. Georges, "Stray light characterization with ultrafast time-of-flight imaging," Scientific reports, vol. 11, no. 1, pp. 1–9, 2021.

[4] L. Clermont, C. Michel, and Y. Stockman, "Stray light correction algorithm for high performance optical instruments: The case of metop-3mi," Remote Sensing, vol. 14, no. 6, p. 1354, 2022.

[5] F. J. Ávila, J. Ares, M. C. Marcellán, M. V. Collados, and L. Remón, "Iterative-trained semi-blind deconvolution algorithm to compensate straylight in retinal images," Journal of Imaging, vol. 7, no. 4, 2021.

[6] Y. Wu, Q. He, T. Xue, R. Garg, J. Chen, A. Veeraraghavan, and J. T. Barron, "How to train neural networks for flare removal," in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 2239–2247.

[7] A. V. Shoshin and E. A. Shvets, "Veiling glare removal: synthetic dataset generation, metrics and neural network architecture," vol. 45, no. 4, pp. 615–626, 2021.

[8] Y. Dai, C. Li, S. Zhou, R. Feng, and C. C. Loy, "Flare7k: A phenomenological nighttime flare removal dataset," in Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track, 2022.

[9] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," CoRR, vol. abs/1505.04597, 2015.

[10] Mittal, Sparsh and Gupta, Saket and Dasgupta, Sudeb "FPGA: An efficient and promising platform for real-time image processing applications", National Conference On Research and Development In Hardware Systems (CSI-RDHS), 2008.

[11] N. Beheshti and L. Johnsson, "Squeeze u-net: A memory and energy efficient image segmentation network," in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops, 2020, pp. 364–365.

[12] G. Bahl, L. Daniel, M. Moretti, and F. Lafarge, "Low-power neural networks for semantic segmentation of satellite images," in Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops, Oct 2019.

[13] S. Lahti, P. Sjövall, J. Vanne, and T. D. Hämäläinen, "Are we there yet? a study on the state of high-level synthesis," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 38, no. 5, pp. 898–911, 2019.

[14] Y.-L. Lin, "Recent developments in high-level synthesis," ACM Trans. Des. Autom. Electron. Syst., vol. 2, no. 1, p. 2–21, jan 1997.

[15] P. Young, A. Lai, M. Hodosh, and J. Hockenmaier, "From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions," Transactions of the Association for Computational Linguistics, vol. 2, pp. 67–78, 02 2014.

[16] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," CoRR, vol. abs/1704.04861, 2017.