Hardware development of skylight estimation processing in haze removing using high-level synthesis

Daiki Shirai^{a,*}, Akira Yamawaki^b

^aKyushu Institute of Technology, 1-1 Sensui-cho, Kitakyushu-shi, Fukuoka, 804-8550 Japan

*Daiki Shirai: shirai.daiki885@mail.kyutech.jp

Abstract

When monitoring infrastructure by using a drone, the real time advanced image processing on the drone can realize self-flying inspection. This result can reduce the burden of off-line processing to image data captured by the drone and realize quicker and more efficient infra monitoring method. In battery-powered embedded devices such as drones, hardware implementation is effective for realizing low-power and high-performance image processing. In addition, the high-level synthesis automatically converting the software to the hardware module can handle different kinds of drones and image processing algorithms quickly and flexibly. Considering the aerial image of the drone, it is important to remove haze from the captured image from the viewpoint of noise removal for subsequent advanced image processing. In this paper, we propose a software description method for high-level synthesis in considering of hardware implementation of skylight estimation processing in haze removal processing by Kaiming He et al^(1,2). In addition, we evaluate the performance of hardware, and show the effect of the proposed method.

Keywords: HLS, pipeline, haze removal

1. Introduction

In recent years, many embedded devices using image recognition have been developed, and the market size of drone is growing rapidly.

Among them, those using aerial images taken by cameras mounted on UAV devices such as drones are attracting particular attention, and research and projects related to them are being widely conducted. In Japan, where infrastructure equipment is aging significantly, the number of cases where equipment is inspected using aerial images

from drones is increasing as the working population declines.

In such cases, the aerial image becomes hazy due to the weather and shooting conditions, making it difficult to perform an accurate inspection.

In order to solve such a problem, we thought that it would be effective to install an image processing module that removes the haze generated in the captured image in real time on the drone, and decided to work on the development of that module.

The module to be developed in this paper is the hardware that executes the skylight estimation processing unit in the haze removal processing by He et al., And high-level synthesis is used for its development. Therefore, we will also develop a software description method that takes high-level synthesis into consideration.

Finally, the hardware obtained by high-level synthesis is compared with pure software that conforms to the original algorithm and software that considers high-level synthesis, and performance is evaluated based on power consumption and execution time during processing. It shows that the development module can perform high-speed processing with low power consumption.

2. Haze Removal Process

2.1 Haze Removal Processing by He et al.

This section describes the haze removal processing algorithm by He et al., Which is dealt with in this paper. Normally, when taking a picture with a camera, light such as the sun (hereinafter referred to as skylight in this paper) hits objects, and the light called direct light that bounces off the object reaches the camera to take an image. On the other hand, when there is haze at the time of shooting, the skylight is scattered in the haze (this is called airglow) and mixes with the airglow in the haze before the direct light reaches the

camera. It would be a different light. As a result, haze occurs in the captured image.

Based on these facts, a model is given by the following equation

$$I(x) = J(x)t(x) + A(1 - t(x)). \quad (1)$$

Here, x is the entire shooting scene, J is the direct light from the object (the original clear image), A is the airlow (assumed to be constant throughout the shooting scene), and $t(0 < t \leq 1)$ is the transmittance (mixing of direct light and skylight), $A(1 - t(x))$ is airlow, and I is the captured image (hazed image).

In order to obtain the original image, the above equation is transformed into the following equation

$$J(x) \approx \frac{I(x) - \tilde{A}(x)}{\tilde{t}(x)} + \tilde{A}(x). \quad (2)$$

Here, $\tilde{A}$ is an estimated value of skylight, and $\tilde{t}$ is an estimated value of transmittance.

From the above, by obtaining $\tilde{A}$ and $\tilde{t}$, direct light from objects, that is, the original clear image can be obtained.

2.2 Software Design for Skylight Estimation

The haze removal process of He et al. is divided into four stages: extraction of skylight region, calculation of the skylight estimate, calculation of the transmittance, and restoration of direct light. This section describes a pure software program that estimates the skylight region and calculates the estimated skylight up to the second stage, which is the target of hardware conversion.

In the estimation of the skylight region, which is the first step, it is assumed that the influence of skylight in the image is reflected in the brightness, and the brightness (Y) is calculated from the R , G , and B values of each pixel (hereinafter referred to as RGB values). In addition, a minimization filter is applied to each local area on the image called a patch. The minimum brightness value in the patch obtained by this filter is compared with an arbitrary threshold value set so as not to cause discomfort in the restored image, and if it is larger than the threshold value, the patch is set as the skylight region.

In the calculation of the skylight estimation value in the second stage, it is assumed that the skylight value in the image is constant, so the skylight estimation value is calculated by averaging the RGB values of the estimated skylight region.

Based on the above, Figure 1 shows a flowchart of a software program for skylight estimation processing.

Fig. 1. Flowchart of pure software for skylight estimation

By this flowchart, we create a software program that executes each 2 processes, and call this pure software.

3. Skylight Estimation Software for HLS

3.1 High-Level Synthesis (HLS)

This section describes the High-Level Synthesis (HLS) used in hardware development and points to note when using it.

HLS is the hardware design method using the design tools and compiler that automatically generates an RTL for hardware whose operation is described in software programming languages.

The design method is so convenient that there is a big difference in the burden on the developer and the development period between the case where HLS is used and the where it is not used.

On the other hand, there are some things to keep in mind when using HLS. The most important thing to pay attention is how to write a software program to be input to the HLS tool. If the software program does not consider HLS, the hardware output from the tool may be unnecessarily large and slow. As a result, the hardware is inferior performance to the software input to the HLS tool.

Therefore, based on the above contents, the pure software presented in Section 2.2 will be improved to software that considers HLS.

3.2 Software Design Considering HLS

This section describes the description method required to change the pure software program created in Section 2.2 to a software program that considers HLS.

First, Figure 2 shows a diagram of the flow of skylight estimation processing for HLS.

(b) Processing flow for High-Level Synthesis

Fig. 2. Pure software processing flow and high-level synthesis software processing flow

Based on this, we will make changes to pure software, paying attention to three major points.

The first point is pixel processing of patch processing. When hardware accesses (reads and writes) memory according to its specifications, it is appropriate to access memory with consecutive addresses. In the processing performed for each patch in the pure software of skylight estimation processing, a discontinuous address is accessed in the memory.

Therefore, by estimating the skylight region for each pixel instead of for each patch, the hardware obtained by

HLS is changed so that it can access continuous memory address.

The second is the deletion of the skylight region extraction map.

In the pure software of skylight estimation processing, the estimation of the skylight region and the calculation of the skylight estimation value are created as independent functions, so the skylight region estimated in the first stage is transferred to the subsequent processing. A skylight region extraction map was created for use and was used as an intermediary.

In the skylight estimation processing software program for HLS to be developed, these two processes can be performed simultaneously for the same pixel, so they are summarized as one function. As a result, in the processing flow for HLS, it is possible to omit the creation of the skylight region extraction map. In addition, since the number of memory accesses can be reduced when hardware is used by HLS, it is possible to greatly improve the performance of development hardware that does not have a cache memory.

The third is the pipeline of processing.

Pipeline processing is a method in which processing in each stage is made independent and processing is executed in parallel for repeated processing (loop) in several stages.

In the hardware to be developed, the processing in the loop can be configured as an independent circuit, so that an ideal pipeline can be applied to the entire target processing, and the processing speed can be improved.

Based on the above, we created a skylight estimation processing software (hereinafter referred to as HLS software) program for HLS.

4. Experiments and Discussions

4.1 Experiment Environment

We input the HLS software program created in Section 3.2 to the Xilinx high-level synthesis tool Vivado HLS 2018.3 to perform high-level synthesis. Subsequently, we mounted the HDL program generated by HLS on the FPGA board ZYBO (Zynq-7000 Development Board) made by DIGILENT by the FPGA development environment software Vivado 2018.3 made by Xilinx, and verified the operation. In addition, in order to confirm the hardware obtained by HLS (hereinafter referred to as HLS hardware) and the data output by the HLS software, we used HP ProDesk 400 G5 SFF (hereinafter referred to as PC), and serial communication to send and receive data between

ZYBO and PC.

Figure 3 shows the relationship between the system configuration deployed on ZYBO and peripheral devices.

The operating frequency of the hardware developed this time is 100MHz, and the operating frequency of the embedded CPU (hereinafter referred to as ZYBO CPU) on the board is 650MHz.

The size of the input image used in the experiment is 1024 x 768 pixels.

Fig. 3. System configuration

4.2 Experiment Environment

In this experiment, the performance of the developed HLS hardware will be evaluated based on the processing performance improvement ratio and the run-time power improvement ratio for the HLS software. In the experiment, when the software is operated, the processing is executed by the ZYBO CPU.

The performance improvement ratio is calculated by calculating the ratio between the software execution time on the ZYBO CPU and the hardware execution time on the FPGA^(3, 4).

The run-time power improvement ratio is calculated by multiplying the performance improvement ratio by the power improvement ratio, which is the ratio of the software operating frequency to the hardware operating frequency^(3, 4).

4.3 Experimental Results and Discussion

Figure 4 (a) shows a graph showing the execution time of each of the HLS software and HLS hardware when the skylight estimation process is executed.

The execution time of the skylight estimation process was 28.05 [ms] for the HLS software and 7.87 [ms] for the HLS hardware, and the processing performance was improved by about 3.2 times. It is considered that this is because the data that can be processed per clock has increased due to the pixel processing and pipeline conversion

(a) Execution time of skylight estimation processing of HLS software and HLS hardware

(b) Run-time power improvement ratio in skylight estimation processing

Fig. 4. Execution time and execution power

improvement ratio of sky light estimation processing

in the HLS hardware.

In addition, the developed HLS hardware was able to achieve a run-time power improvement of approximately 23.1 times that of software (Figure 4 (b)).

However, if the skylight estimation process is executed using the HLS hardware for the entire haze removal process, the software execution time for the subsequent calculation of the estimated transmittance and the restoration of the clear image is long, and the benefits of this HLS hardware are hidden. It turned out that it would end up.

5. Conclusion

For the development of low power consumption and high-speed haze removal hardware, HLS has developed the hardware for skylight estimation processing, which is the pre-processing.

By applying various ideas from the conventional algorithm, we were able to develop skylight estimation

processing software for HLS. However, it turns out that the effect is hidden from the perspective of haze removal as a whole.

As a future task, we will develop the hardware for the remaining two processes in the same way, and aim to make the entire haze removal hardware, thereby further improving the processing performance and power efficiency.

References

- (1) K. He et al: "Single Image Haze Removal Using Dark Channel Prior. Proc.", IEEE CVPR, Vol.1, pp.1956-1963, 2009
- (2) Hiroaki Kotera: "A color Correction for Degraded Scenes by Air Pollution", Journal of the Color Science Association of Japan, Vol.40, No.2, pp. 49-59, 2016
- (3) Moena Yamasaki, Satoshi Yonemitsu and Akira Yamawaki: "Effect of Redundant Function Execution to Reduce Memory Access on High-level Synthesis", Processing of the 6th IIAE International Conference on Industrial Application Engineering 2018, pp.206-209, 2018
- (4) Moena Yamasaki and Akira Yamawaki: "Description Method for High-level Synthesis of Histogram Generation and Their Evaluation", IEIE Transactions on Smart Processing and Computing, Vol. 8, No. 3, pp.178-185, 2019