Computational Cost Analysis and Implementation of Accelerated Iterative Shrinkage Smoothing

Dabwitsu Kasauka†, Hiroshi Tsutsui†, Hiroyuki Okuhata*, and Yoshikazu Miyanaga†
†Graduate School of Information Science and Technology, Hokkaido University
Kita 14, Nishi 9, Kita-ku, Sapporo, Hokkaido 060-0814, Japan
*Synthesis Corporation
Awajimachi TC Bldg. 4th Floor, Awajimachi 2-6-9, Chuo-ku, Osaka 541-0047, Japan

Abstract—In this paper, we present a computational cost analysis result of accelerated iterative shrinkage smoothing algorithm, which is one of promising image smoothing algorithms with sufficient smoothing quality results and reduced processing time. The main motivation of this cost analysis is to provide a base for efficient hardware implementation. We implemented it in a lower-level programming language with OpenCV library as opposed to the MATLAB implementation. The resolution dependency of the processing time is also illustrated.

I. INTRODUCTION

Digital image smoothing research has become increasingly popular in recent years. A good smoothing algorithm should have the ability of reducing noise at same time, preserving important details, such as edge. Edge-preserving smoothing can be achieved via energy-minimizing, surface fitting, and weighted averaging [1]. Both spatial and frequency domain techniques can be utilized for smoothing an image. While spatial domain techniques are generally faster, frequency domain techniques are more efficient when dealing with large kernels.

Image smoothing can be involved in a wide range of applications from lower-level ones such as image enhancement and noise reduction to higher-level ones such as image recognition and segmentation. In the case of image enhancement such as detail enhancement and high dynamic range (HDR) tone mapping, real-time processing is sometimes required to process and display video data. Therefore an algorithm with sufficient smoothing quality results and reduced processing time is strongly demanded.

The smoothing algorithm proposed in [2] is our point of focus since it requires fewer iterations to produce suitable results compared to some of the most recent algorithms such as [3]. Apart from relatively lower processing time, it does not result in unwanted blur or artifacts.

Motivated by this, in this paper, we analyze the computational cost of the algorithm proposed in [2]. This may provide a base for efficient hardware implementation. Our goal is implementation of real-time video image enhancement such as [4] based on state-of-the-art smoothing algorithms.

This paper is organized as follows. In Section II, we briefly describe the smoothing algorithm proposed in [2]. In Section III, we show the computational cost analysis results with the resolution dependency. Since the algorithm was implemented in both a high-level language (MATLAB) and a lower-level language (C++) with OpenCV library, we show the processing time for each implementation. In Section IV, some discussions toward real-time hardware implementation are shown. Finally, in Section V, we conclude this paper.

II. ALGORITHM OVERVIEW

In this section, we briefly describe the theory involved in fast multi-scale detail decomposition via accelerated iterative shrinkage [2]. The basis of this algorithm is the use of first order proximal operators as a way to accelerate the shrinkage scheme. According to [2], a smoothing problem is formulated as

$$\arg\min_u \frac{\lambda}{2} \|u - g\|^2 + \psi(\nabla u),$$  (1)

where $g$ is input image, $u$ is smoothed output, $\psi(\nabla u)$ is gradient function, and $\lambda$ is a positive regularization term. Problem (1) can be simplified by applying half-quadratic form, and solved by iterative minimization as follows,

$$v^{(k+1)} \leftarrow \arg\min_v \psi(v) + \frac{\beta}{2} \|\nabla u^{(k)} - v\|^2,$$  (2)

$$u^{(k+1)} \leftarrow \arg\min_u \|u - g\|^2 + \frac{\beta}{2} \|\nabla u - v^{(k+1)}\|^2.$$  (3)

where $v^{(k+1)}$ corresponds to a shrinkage operation, $u^{(k+1)}$ corresponds to the screened Poisson equation [5], and $\beta$ is the new regularization term. By applying first order proximal operator, $v^{(k+1)}$ can be simplified to

$$v^{(k+1)} \leftarrow \nabla u^{(k)} \circ f \left( \nabla u^{(k)} \right),$$  (4)

where $f$ is the shrinkage weight. For a color image, the following shrinkage weights can be used,

$$f_1(T(\nabla u)) = 1 - \frac{1}{1 + (T(\nabla u)/\gamma)\alpha},$$

$$f_2(T(\nabla u)) = 1 - e^{-(T(\nabla u)/\gamma)\alpha},$$  (5)

where $\alpha$ and $\gamma$ are positive parameters, and $T(\nabla u)$ is given by

$$T(\nabla u) = \sqrt{\left( \sum_{k=1}^{ch} \frac{\partial u_k}{\partial x} \right)^2 + \left( \sum_{k=1}^{ch} \frac{\partial u_k}{\partial y} \right)^2},$$  (6)
where $u_k$ corresponds to a color channel of $u$. From Eq. (3), $u^{(k+1)}$ is solved using fast Fourier transform and given by

$$u^{(k+1)} \leftarrow \mathcal{F}^{-1} \left( \frac{\mathcal{F}(\lambda g - \beta \text{div}(v^{(k+1)}))}{\lambda - \beta \text{lap}} \right),$$

(7)

where $\text{div}$ is the discrete divergence, and $\mathcal{F}$ and $\mathcal{F}^{-1}$ are forward and inverse fast Fourier transforms, respectively.

In Eq. (7), lap is independent of the input image. It is basically the optical transfer function of a discrete Laplacian filter such as

$$
\begin{pmatrix}
0 & 1 & 0 \\
1 & -4 & 1 \\
0 & 1 & 0
\end{pmatrix}.
$$

(8)

Finally, an initial solution calculation prior to the smoothing iteration is performed as a way to speed up the smoothing process [2]. This is approximated by

$$u^{(0)} \approx g + \xi \text{div}(\nabla g - \nabla \circ f_i(T(\nabla g))).$$

(9)

According to [6], if the luminance component and color component of a color image can be respectively described, not only more useful information can be gained from the color image, but also the processing technologies used to process the color image. Hence, in this algorithm, smoothing is performed in CIELab colorspace.

We summarize the algorithm flow as shown in Fig. 1. The detail flow of the stage of smoothing iterations is shown in Fig. 2. Figure 4 is a sample smoothing result of an input image shown in Fig. 3 using the algorithm [2].

### III. Computational Cost Analysis

In this section, we show the computational cost analysis results with the resolution dependency. The experiments were performed on Intel Core i7 CPU @3.4 MHz. The number of iterations was 2 as proposed in [2]. Table I shows the processing time breakdown of the algorithm for various image resolutions. Figure 5 represents the same data as that of Table I in graphical presentation.

As is evident from the overall distribution in Fig. 5, the smoothing iteration stage is the most computational costly stage. Below, we investigate this stage further in order to have a clearer view of what constitutes it.

The detailed breakdown of the processing time of the iteration stage represented is given in Table II and Fig. 6,
### TABLE I
PROCESSING TIME BREAKDOWN FOR VARIOUS RESOLUTIONS.

<table>
<thead>
<tr>
<th>Image label and resolution</th>
<th>Image label and resolution</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-processing (sec)</td>
<td>0.0121</td>
<td>0.1452</td>
<td>0.2541</td>
<td>0.5241</td>
<td></td>
</tr>
<tr>
<td>Initial solution (sec)</td>
<td>0.0039</td>
<td>0.0954</td>
<td>0.2046</td>
<td>0.4112</td>
<td></td>
</tr>
<tr>
<td>Filter lap (sec)</td>
<td>0.0015</td>
<td>0.0423</td>
<td>0.1027</td>
<td>0.2049</td>
<td></td>
</tr>
<tr>
<td>Iteration stage (sec)</td>
<td>0.0244</td>
<td>0.3971</td>
<td>1.3181</td>
<td>5.3685</td>
<td></td>
</tr>
<tr>
<td>Final processing (sec)</td>
<td>0.0011</td>
<td>0.0121</td>
<td>0.0215</td>
<td>0.0614</td>
<td></td>
</tr>
<tr>
<td>Overall time (sec)</td>
<td>0.0430</td>
<td>0.8921</td>
<td>1.9010</td>
<td>4.5701</td>
<td></td>
</tr>
</tbody>
</table>

### TABLE II
PROCESSING TIME BREAKDOWN OF THE ITERATION STAGE FOR VARIOUS RESOLUTIONS.

<table>
<thead>
<tr>
<th>Image label and resolution</th>
<th>Image label and resolution</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gradient and shrinkage calc. (sec)</td>
<td>0.0046</td>
<td>0.1310</td>
<td>0.2990</td>
<td>0.8190</td>
<td></td>
</tr>
<tr>
<td>Color channel separation (sec)</td>
<td>0.0001</td>
<td>0.0051</td>
<td>0.0124</td>
<td>0.0248</td>
<td></td>
</tr>
<tr>
<td>DFT (sec)</td>
<td>0.0696</td>
<td>0.1764</td>
<td>0.3084</td>
<td>1.0734</td>
<td></td>
</tr>
<tr>
<td>Smoothing (sec)</td>
<td>0.0054</td>
<td>0.1328</td>
<td>0.3294</td>
<td>0.3318</td>
<td></td>
</tr>
<tr>
<td>IDFT (sec)</td>
<td>0.0035</td>
<td>0.1344</td>
<td>0.2416</td>
<td>0.3986</td>
<td></td>
</tr>
<tr>
<td>Color channel merge (sec)</td>
<td>0.0001</td>
<td>0.0040</td>
<td>0.0082</td>
<td>0.0196</td>
<td></td>
</tr>
<tr>
<td>Total (sec)</td>
<td>0.0853</td>
<td>0.5837</td>
<td>1.1980</td>
<td>3.1772</td>
<td></td>
</tr>
</tbody>
</table>

![Fig. 5. Processing time distribution and percentage of total.](image1)

![Fig. 6. Iteration stage’s processing time distribution and percentage of total.](image2)
for the same image resolutions shown in Table I.

From this, the collective Fourier transform process and the smoothing calculations according to Eq. (7) attributes most of the processing time of this stage. Clearly, solving the screened Poisson equation $u^{(k+1)}$ given in Eq. (3) using Fourier transform has a great impact on the processing time.

Figure 7 illustrates the relationship between image resolution and processing time for both MATLAB and OpenCV implementation. From Fig. 7, it can be said the processing time of this algorithm has an approximately linear relationship between the image resolution and the processing time. In addition, we confirmed that implementing it in a lower-level language with OpenCV library led to approximately 45% reduction in the processing time as compared to the MATLAB implementation.

IV. DISCUSSIONS

From Fig. 2, it can be noted as illustrated by the internal loop that the smoothing process according to Eq. (7) is performed for each color channel consecutively. Therefore, in an effort to reduce the processing time in a real-time application, it can be proposed to process the color channels simultaneously by implementing parallel processing. In the case of a 2 Mpixel color image (image label 3 whose resolution is 1,080×1,920), according to Fig. 2, per-color channel processing (DFT, smoothing, and IDFT) takes approximately 0.29 seconds, while approximately 0.88 seconds for all collectively. Hence by parallel processing, approximately 0.59 seconds can be cut down on. This would reduce the overall processing time in this case from 1.9 seconds to about 1.3 seconds, which is a 31% improvement in processing time.

Several approaches can be applied to reduce the the processing time for high resolution images. One approach is translation of Eq. (7), where frequency domain processing is performed, to special domain processing. Considering a hardware implementation, DFT and IDFT have great impact on not only processing time but also memory usage due to pixel data dependency. Therefore, if the data dependency can be successfully reduced utilizing special domain processing, it contributes to both processing time and memory usage reduction. Another approach is application of image scaling into the smoothing algorithm as shown in Fig. 8. Although a trade-off between processing time and smoothing quality is likely to occur, a suitable scaling technique or a post-processing referring the input image can be used to mitigate the degree of trade-off.

V. CONCLUSION

According to the analysis of [2], we identify that the Fourier transform involved in the smoothing iteration stage contributes greatly to the overall processing time. In addition, processing the color channels simultaneously would considerable improve the processing time performance. In this paper, we confirmed that the processing time of the algorithm proposed in [2] is approximately linearly dependent on the resolution of the input image. As one of future works, we are planning to develop an algorithm which alleviates the high computational due to the Fourier transform. Developing an algorithm suitable for VLSI implementation is another future work.

REFERENCES