IEICE TRANSACTIONS

on Fundamentals of Electronics, Communications and Computer Sciences

DOI:10.1587/transfun.2022VLP0009

Publicized:2022/09/05

A PUBLICATION OF THE ENGINEERING SCIENCES SOCIETY

The Institute of Electronics, Information and Communication Engineers

Kikai-Shinko-Kaikan Bldg., 5-8, Shibakoen 3 chome, Minato-ku, TOKYO, 105-0011 JAPAN

PAPER Special Section on VLSI Design and CAD Algorithms

Real-Time Image-Based Vibration Extraction with Memory-Efficient Optical Flow and Block-Based Adaptive Filter

Taito MANABE[†a)], ***Nonmember and*** **Yuichiro SHIBATA**[†b)], ***Member***

SUMMARY In this paper, we propose a real-time vibration extraction system, which extracts vibration component within a given frequency range from videos in real time, for realizing tremor suppression used in microsurgery assistance systems. To overcome the problems in our previous system based on the mean Lucas-Kanade (LK) optical flow of the whole frame, we have introduced a new architecture combining dense optical flow calculated with simple feature matching and block-based band-pass filtering using band-limited multiple Fourier linear combiner (BMFLC). As a feature of optical flow calculation, we use the simplified rotation-invariant histogram of oriented gradients (RIHOG) based on a gradient angle quantized to 1, 2, or 3 bits, which greatly reduces the usage of memory resources for a frame buffer. An obtained optical flow map is then divided into multiple blocks, and BMFLC is applied to the mean optical flow of each block independently. By using the L^1-norm of adaptive weight vectors in BMFLC as a criterion, blocks belonging to vibrating objects can be isolated from background at low cost, leading to better extraction accuracy compared to the previous system. The whole system for 480p and 720p resolutions can be implemented on a single Xilinx Zynq-7000 XC7Z020 FPGA without any external memory, and can process a video stream supplied directly from a camera at 60 fps.

key words: *Optical Flow, Feature Matching, BMFLC, FPGA, Real-Time*

1. Introduction

Microsurgery, a delicate surgery using a surgical microscope, has been performed commonly. In microsurgery, however, an involuntary movement called tremor is a severe problem to a surgeon. Developing a microsurgery assistance system which detects and suppresses physiological hand tremor by actively generating anti-phase vibration is one of the solutions to this problem.

Various research works have been carried out to date on tremor attenuation from a wide range of aspects [1]–[10]. For isolating the vibration component of tremor from voluntary motion without phase delay, which is an essential part of active tremor canceling, Riviere et al. proposed an adaptive algorithm called the weighted frequency Fourier linear combiner (WFLC) [1]. WFLC estimates a quasi-periodic signal by adjusting amplitude, phase, and frequency of a truncated Fourier series model. The concept of WFLC was extended by Veluvolu et al. as the band-limited multiple Fourier linear combiner (BMFLC) [2] so that a signal containing multiple frequency components can be effectively handled. They executed BMFLC as software on a real-time operating system to process input data from an accelerometer. Rocon et al. implemented a WFLC-based tremor suppression system in a rehabilitation robotic exoskeleton [3], where tremor signals were acquired with gyroscopes.

Though most tremor attenuation systems including these depend on inertial sensors, it is undesirable to put such sensors on top of modern microscopic surgical instruments, as they may disturb delicate works. A promising approach is to use a real-time image processing technique to obtain vibration components in a live video from a stationary microscope. Use of a camera will not be hindrance to surgical procedures, since heads-up surgery using a display showing the microscopic image is very common nowadays. However, many research works using visual information have focused on detecting and analyzing tremor, not on canceling it. For example, Uhrikova et al. implemented an image-based system for measuring the frequency of tremor [7]. Soran et al. developed a tremor detection system using a support vector machine (SVM) in the frequency domain [9], and Wang et al. proposed a hand tremor detection system based on neural-network-based approaches [10], both of which detect the presence of tremor. These are useful for diagnosis purposes, but not directly applicable to real-time tremor canceling.

To generate anti-phase vibration for tremor suppression using an image-based approach, both extraction of vibration components and band-pass filtering must be done in real time with low latency. This is difficult to accomplish with software processing on CPU where the entire frame data must be kept on memory before processing. We have addressed this challenge by using a pipelined architecture using an FPGA which is directly connected to a camera. In [11], we proposed the method combining the mean Lucas-Kanade (LK) optical flow [12] of the whole frame and BMFLC adaptive filtering. Though the system based on this method achieved low latency, it could not extract large vibration components accurately and was severely affected by optical flow of the background, as described later in Sect. 2. To overcome these problems, we propose a tremor-extraction system with a completely-redesigned algorithm in this paper. The major contributions of this paper include:

- Simple feature matching using a quantized gradient angle enables fully-pipelined optical flow calculation for 720p videos without any external memory.
- Block-based BMFLC filtering enables detection of the region belonging to a vibrating object at low cost, leading to better extraction accuracy.

[†]The authors are with Nagasaki University, Nagasaki-shi, 852-8521, Japan.
a) E-mail: tmanabe@nagasaki-u.ac.jp
b) E-mail: shibata@cis.nagasaki-u.ac.jp

- Deeply-pipelined FPGA design and implementation of the proposed approach are presented for real-time vibration extraction at 60 fps with low latency.

The rest of this paper is organized as follows. We firstly make a brief explanation on our previous system in Sect. 2. Based on this, the newly-proposed algorithm and its implementation are detailedly described in Sect. 3 and 4, respectively. Sect. 5 shows evaluation results of the system, and finally, we conclude the paper in Sect. 6.

2. Overview of the Previous System

To clarify the advantage of the newly-proposed algorithm, we describe the overview of the algorithm and implementation of the previous system [11] as well as its problems.

2.1 Optical Flow Estimation with LK method

First, a dense Lucas-Kanade (LK) optical flow is calculated. Unlike the methods using an iterative approach like the Horn-Schunck method [13], the plain LK method is simple and easy to implement on hardware. Let the pixel value (luminance) at the coordinates (x, y) at the time t be $I(x, y, t)$. Given that the object located at (x, y) at the time t moves to $(x+\Delta x, y+\Delta y)$ at the time $t+\Delta t$, and the pixel value remains unchanged, then the following relation holds:

$$I(x,\ y,\ t) = I(x + \Delta x,\ y + \Delta y,\ t + \Delta t) \tag{1}$$

On the assumption that the image is differentiable and $\Delta x, \Delta y$ are small, the following approximate expression can be obtained by Taylor-expanding Eq. (1) to the 1st order:

$$I_x u + I_y v + I_t \simeq 0$$
$$\left(I_x = \frac{\partial I}{\partial x},\ I_y = \frac{\partial I}{\partial y},\ I_t = \frac{\partial I}{\partial t},\ u = \frac{\Delta x}{\Delta t},\ v = \frac{\Delta y}{\Delta t}\right) \tag{2}$$

where I_x, I_y are the horizontal and vertical intensity gradients, I_t is the difference of the pixel values between the 2 frames, and (u, v) is the optical flow at (x, y). By introducing the assumption that neighboring pixels in a small region have the same optical flow, we can estimate (u, v) using the least squares method.

In the previous system, this algorithm is implemented as a fully-pipelined structure to estimate optical flow. However, because of the assumption that $\Delta x, \Delta y$ are small, large optical flow cannot be extracted accurately. The common solution to this is the introduction of the pyramidal approach, which we did not adopt since it makes the structure and control of the pipeline complicated. In addition, the assumption of the luminance invariance also makes the system vulnerable to a change in light condition. Large memory requirement for a frame buffer is also problematic in embedded platforms.

2.2 Mean Optical Flow Calculation

By taking the average of all the optical flow values in a frame for both horizontal and vertical directions, the mean optical flow of the frame $(\overline{u}, \overline{v})$ are calculated. To mitigate the effect from background (the region outside a surgical tool) and noises, only the optical flow whose absolute value is greater than the pre-defined threshold is taken into account. The sign filtering, which ignores negative or positive values when the sum of all the values is positive or negative, is applied at the same time. However, these mechanisms work only when the background stands still. The fact that they only consider the movement (displacement) from the previous frame is also inappropriate for the filtering of vibration component.

2.3 BMFLC Filtering

The band-pass filtering using BMFLC is applied to $\overline{u}$ and $\overline{v}$. In BMFLC, the pass frequency band $[f_{lower}, f_{upper}]$ is equally divided into L sub-bands, and the reference input vector $\boldsymbol{x}$ is generated based on the current time t [sec]:

$$\boldsymbol{x} = (\sin\omega_0 t,\ \ldots,\ \sin\omega_{L-1} t, \cos\omega_0 t,\ \ldots,\ \cos\omega_{L-1} t)^T$$
$$\left(\omega_r = 2\pi\left(f_{lower} + \frac{f_{upper} - f_{lower}}{L} r\right)\right) \tag{3}$$

Considering the characteristics of physiological tremor [14], we set f_{lower} and f_{upper} to 8 and 12, respectively. The number of harmonics L is set to 16.

We can estimate the band-limited values of the input signals $\overline{u}$ and $\overline{v}$ by taking the weighted sums of $\boldsymbol{x}$ and the adaptive weight vectors $\boldsymbol{w}_x = (w_{x,0}, w_{x,1}, \ldots, w_{x,2L-1})^T$ and $\boldsymbol{w}_y = (w_{y,0}, w_{y,1}, \ldots, w_{y,2L-1})^T$, as follows:

$$(\Delta x, \Delta y) = (\boldsymbol{w}_x^T \boldsymbol{x}, \boldsymbol{w}_y^T \boldsymbol{x}) \tag{4}$$

The adaptive weight vectors $\boldsymbol{w}_x$ and $\boldsymbol{w}_y$ are updated using the least mean square algorithm [15], as follows:

$$\boldsymbol{w}_x \leftarrow \boldsymbol{w}_x + 2\mu(\overline{u} - \Delta x)\boldsymbol{x} \tag{5}$$
$$\boldsymbol{w}_y \leftarrow \boldsymbol{w}_y + 2\mu(\overline{v} - \Delta y)\boldsymbol{x} \tag{6}$$

The gain parameter $\mu = 2^{-7}$ controls the balance between convergence speed and stability.

By repeating the 3 steps described in Sect. 2.1 to 2.3 for each frame, the extracted vibration components $(\Delta x, \Delta y)$ can be obtained. Note that all calculations are done using integer or fixed-point arithmetic to reduce resource utilization.

3. Algorithm

As explained in Sect. 2, our previous vibration extraction system had several problems including low accuracy of optical flow estimation, large memory utilization, and insufficient isolation between background and target area. To cope with these problems, we have designed a new vibration extraction algorithm. The algorithm can be divided into 6 steps, each of which is described detailedly in this section.

3.1 Gradient Calculation

Firstly in this step, the 3×3 Gaussian kernel G is applied

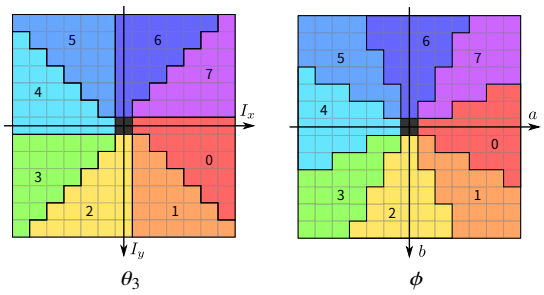


Fig. 1 Calculation of 3-bit gradient angle

to the input 8-bit grayscale image I to obtain the smoothed image $I' = I * G$. Next, the horizontal and vertical gradients I_x and I_y are calculated as follows:

$$I_x(x, y) = I'(x + 1, y) - I'(x - 1, y) \tag{7}$$
$$I_y(x, y) = I'(x, y + 1) - I'(x, y - 1) \tag{8}$$

where x and y are the horizontal and vertical coordinates with the origin placed at the top-left edge of the image. Based on the gradient values, the gradient angle θ is calculated for each pixel. To save memory resources for a frame buffer, θ is quantized to 1, 2, or 3 bits. Firstly, the 3-bit gradient angle θ_3 $(0 \le \theta \le 7)$, as shown in Fig. 1, is calculated:

$$\theta_3(x, y) = \left\lfloor \frac{4 \times \mathrm{atan2}(I_y(x, y), I_x(x, y))}{\pi} \right\rfloor \bmod 8 \tag{9}$$

θ_3 can be calculated using the simple magnitude correlation of I_x and I_y, making it robust to brightness change.

If the quantization bit width $Q \in \{1, 2, 3\}$ is 1 or 2, θ_3 is further quantized with spatial dithering by adding the offset e depending on the coordinates. As a result, θ $(0 \le \theta \le 2^Q - 1)$ is represented as follows:

$$\theta(x, y) = \left\lfloor \frac{\theta_3(x, y) + e(x, y)}{2^{3-Q}} \right\rfloor \tag{10}$$
$$e(x, y) = (x + y) \bmod 2^{3-Q} \tag{11}$$

Examples of the quantization for each bit width is shown in the first row of Fig. 2. The 3-bit gradient angle map shown on the left is quantized to 2 bits or 1 bit using the dithering pattern. With 1-bit quantization, memory utilization for a frame buffer can be reduced to 1/8.

3.2 Feature Calculation

For each of the current and previous frames, a simple feature used for optical flow estimation is generated from the quantized gradient angle θ. The algorithm is based on the rotation-invariant histogram of oriented gradients (RIHOG) [16], a variant of the histogram of oriented gradients (HOG) [17]. HOG is a well-known feature descriptor based on a histogram of gradient orientation, which is robust to illumination change. The first step of feature generation is to dequantize the θ to get a 3-bit gradient angle θ':

$$\theta'(x, y) = (2^{3-Q}\theta(x, y) - e(x, y)) \bmod 8 \tag{12}$$

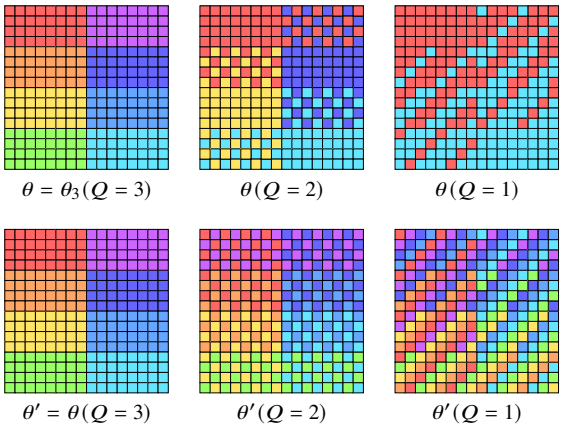


Fig. 2 Quantization and dequantization of gradient angle

This is also illustrated in the second row of Fig. 2. Then, for each pixel, the simplified RIHOG feature is calculated from the small region (cell) of size $C \times C$ surrounding the pixel. Let us consider the relative coordinates (a, b) in a cell, where the center of the cell is regarded as the origin. Then, to make the feature rotation-invariant, the relative gradient angle θ_{rel} of each pixel in the cell is calculated as follows:

$$\theta_{rel}(x, y, a, b) = (\theta'(x + a, y + b) - \phi(a, b)) \bmod 8 \tag{13}$$

Here, $\phi(a, b)$ is the 3-bit angular coordinates of each point in the cell. As shown in Fig. 1, ϕ is calculated in the almost same way as θ_3, except that the offset of $\pi/8$ is added:

$$\phi(a, b) = \left\lfloor \frac{4 \times \mathrm{atan2}(b, a)}{\pi} + \frac{1}{2} \right\rfloor \bmod 8 \tag{14}$$

Note that ϕ and θ_{rel} is undefined when $a = b = 0$.

Finally, the simplified RIHOG feature is calculated by making a histogram of the relative gradient angles in the cell. Let the number of pixels with $\theta_{rel} = i$ $(0 \le i \le 7)$ be n_i, then the feature descriptor H is represented as:

$$H(x, y) = (n_0, n_1, n_2, ..., n_7)^T \tag{15}$$
$$\left(0 \le n_i \le C^2 - 1,\ \sum_{i=0}^{7} n_i = C^2 - 1\right)$$

The distribution of n_i depends on the quantization bit width Q. If Q is 1 or 2, the peak number of n_i gets smaller since the dithering works as a smoothing filter. The possible maximum value of n_i is expected to be $(C^2 - 1)/2$ for $Q = 2$ and $(C^2 - 1)/4$ for $Q = 1$, indicating that the feature can be represented with less bits. However, because of the normalization using ϕ, the actual value can exceed these limits. The offset to ϕ shown in Eq. (14) helps to mitigate this overflow by avoiding the resemblance to the dithering pattern described in Sect. 3.1.

The flow of the feature calculation is illustrated in Fig. 3. Unlike the common HOG feature, the gradient magnitude is not considered. This enables us not only to make a frame

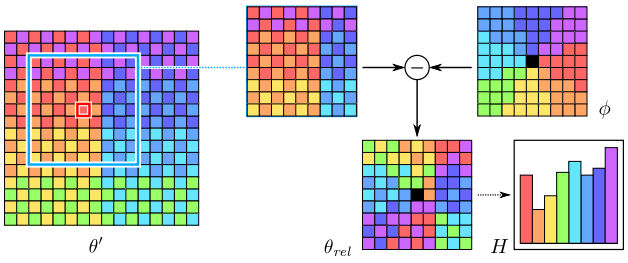


Fig. 3 Feature calculation

buffer small but also to remove the need for histogram normalization with accumulation and division.

3.3 Optical Flow Estimation Using Feature Matching

Based on the feature vector of the current frame H_k and that of the previous frame H_{k-1}, optical flow of each pixel is estimated using pattern matching. Given that the optical flow at the point (x, y) is $F(x, y) = (u, v)$, the point (x, y) in the current frame corresponds to the point $(x - u, y - v)$ in the previous frame. Here, we define the feature similarity $S(u, v)$ using the histogram intersection (HI) [18]:

$$S(u, v) = \max(S_{raw}(u, v) - p(u, v), 0) \tag{16}$$

$$S_{raw}(u, v) = HI(H_k(x, y), H_{k-1}(x - u, y - v)) \tag{17}$$

$$p(u, v) = \left\lfloor \frac{\sqrt{u^2 + v^2}}{2} + 0.5 \right\rfloor \tag{18}$$

The histogram intersection of two histograms $H_1 = (n_{1,i})$ and $H_2 = (n_{2,i})$ is defined as:

$$HI(H_1, H_2) = \sum_i \min(n_{1,i}, n_{2,i}) \tag{19}$$

And $p(u, v)$ is a penalty term to mitigate noises. Using the similarity, the best (u, v) in the pre-defined search window of size $K \times K$ that maximizes the similarity is searched for:

$$F(x, y) = \arg\max_{(u,v)} S(u, v) \tag{20}$$

At the same time, the reliability of the optical flow $R(x, y)$ used in the following step is calculated as follows:

$$R(x, y) = \max_{u,v} S(u, v) - \overline{S}, \quad \overline{S} = \frac{\sum_{u,v} S(u, v)}{K^2} \tag{21}$$

This is the difference between the mean similarity and the maximum similarity, playing a role to prioritize the flow with the specifically-high similarity.

3.4 Block-Mean Optical Flow Calculation

Next, the image is divided into small blocks, each of which has the size of $B \times B$, and the mean optical flow of each block is calculated. Let the optical flow and reliability of the j-th pixel in the block i $(0 \le j \le B^2 - 1)$ be $F_i(j) = (u_{ij}, v_{ij})$ and $R_i(j)$, respectively. Then, the block-mean optical flow $\overline{F}_i = (\overline{u}_i, \overline{v}_i)$ is calculated as the weighted average:

$$\overline{F}_i = (\overline{u}_i, \overline{v}_i) = \left(\frac{\sum_j u_{ij} \times R_i(j)}{\sum_j R_i(j)}, \frac{\sum_j v_{ij} \times R_i(j)}{\sum_j R_i(j)} \right) \tag{22}$$

3.5 Per-Block BMFLC Filtering

Unlike the previous system, we apply BMFLC filtering to each of the blocks independently, not to the mean optical flow of the entire frame. Let us denote the adaptive weight vectors of the block i for horizontal and vertical directions as $\boldsymbol{w}_{x,i}$, $\boldsymbol{w}_{y,i}$, respectively. Note that the reference input vector $\boldsymbol{x}$ is shared among all the blocks. Here, the band-limited estimated mean optical flow of block i is represented as:

$$\overline{F}'_i = (\overline{u}'_i, \overline{v}'_i) = (\boldsymbol{w}_{x,i}^T \boldsymbol{x}, \boldsymbol{w}_{y,i}^T \boldsymbol{x}) \tag{23}$$

The adaptive weight vectors are updated as follows:

$$\boldsymbol{w}_{x,i} \leftarrow \boldsymbol{w}_{x,i} + 2\mu(\overline{u}_i - \overline{u}'_i)\boldsymbol{x} \tag{24}$$

$$\boldsymbol{w}_{y,i} \leftarrow \boldsymbol{w}_{y,i} + 2\mu(\overline{v}_i - \overline{v}'_i)\boldsymbol{x} \tag{25}$$

With this block-wise BMFLC filtering, the region belonging to a vibrating object (surgical tool) can be detected using the block weight W_i calculated as follows:

$$W_i = \max\left(\frac{\|\boldsymbol{w}_{x,i}\|^1 + \|\boldsymbol{w}_{y,i}\|^1}{2} - \tau, 0 \right) \tag{26}$$

Basically, this is the mean of the L^1-norms of adaptive weight vectors for both directions. To ignore the blocks with small weights, the threshold τ is subtracted from the mean.

3.6 Vibration Component Calculation

Finally, the vibration component $(\Delta x, \Delta y)$ is extracted as the weighted mean of the estimated mean optical flows $\overline{F}'_i = (\overline{u}'_i, \overline{v}'_i)$ and the block weights W_i of all the blocks:

$$(\Delta x, \Delta y) = \left(\frac{\sum_i \overline{u}'_i \times W_i}{\sum_i W_i}, \frac{\sum_i \overline{v}'_i \times W_i}{\sum_i W_i} \right) \tag{27}$$

This is the output for each input frame.

4. Implementation

The entire system is described using SystemVerilog. An overview of the system is shown in Fig. 4. The system inputs a pixel stream of 8-bit grayscale image with 480p (640×480) or 720p (1280 × 720) resolution, and is synchronized with the pixel clock. In this section, we briefly explain the design and implementation of each module in the system.

4.1 Gradient Calculation Module

The gradient calculation module firstly applies the Gaussian and the differential kernels to the luminance value given

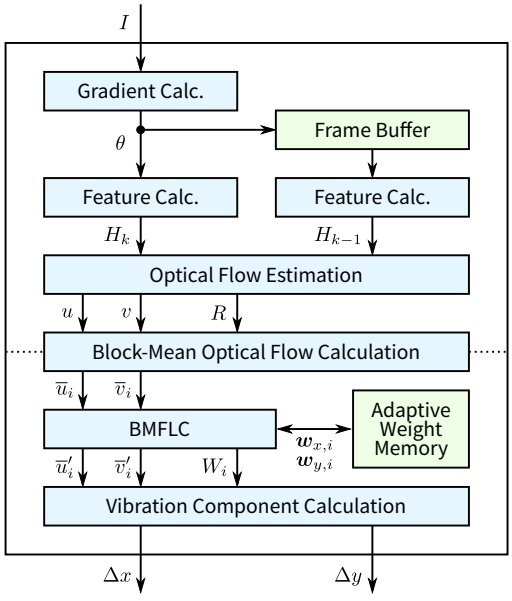


Fig. 4 System Overview

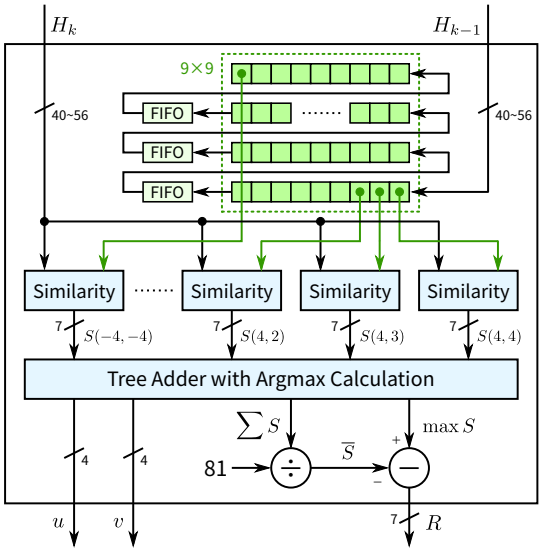


Fig. 5 Overview of Optical Flow Estimation Module

in a raster-scan manner, with the stream processing using a moving window composed of FIFOs and shift registers. Next, the 3-bit gradient angle θ_3 is calculated directly from I_x and I_y using the following equation:

$$(\alpha, \beta, q) = \begin{cases} (I_x, I_y, 0) & (I_x > 0,\ I_y \geq 0) \\ (I_y, -I_x, 1) & (I_x \leq 0,\ I_y > 0) \\ (-I_x, -I_y, 2) & (I_x < 0,\ I_y \leq 0) \\ (-I_y, I_x, 3) & (I_x \geq 0,\ I_y < 0) \end{cases} \tag{28}$$

$$\theta_3 = \begin{cases} 2q & (\alpha > \beta) \\ 2q + 1 & (\text{otherwise}) \end{cases} \tag{29}$$

Eq. (28) corresponds to the quadrant selection, and Eq. (29) divides the quadrant into two. θ_3 is undefined when $I_x = I_y = 0$. Then, θ_3 is further quantized to θ with $Q = 1$, 2, or 3 bits, based on Eq. (10).

4.2 Feature Calculation Module

This module calculates the feature of each pixel in the current and previous frames with the stream processing. First, the 3-bit dequantized gradient angle θ' is calculated using Eq. (12). Next, all the relative gradient angles θ_{rel} in the cell, which is implemented as a moving window, are calculated in parallel based on Eq. (13). And finally, for each of $i = 0, 1, ..., 7$, the number of pixels n_i is calculated using a tree adder to generate the feature $H = \{n_0, n_1, ..., n_7\}$.

Currently, the cell size C is set to 11. Since the value range of n_i is $0 \leq n_i \leq 120 < 2^7$ based on Eq. (15), H can be represented with $7 \times 8 = 56$ bits at most. As explained in Sect. 3.2, this can be reduced when using $Q = 1$ ($5 \times 8 = 40$ bits) or $Q = 2$ ($6 \times 8 = 48$ bits). If an overflow occurs, n_i is clipped to the maximum number allowed.

4.3 Optical Flow Estimation Module

In this module, the optical flow $F = (u, v)$ is estimated based on the feature of the current frame H_k and that of the previous frame H_{k-1}. We set the search window size K to 9. The overview is shown in Fig. 5.

To begin with, all neighboring values of H_{k-1} in the search window of size 9×9 that appear in a moving window are obtained. The latency required here is compensated by reducing the length of the frame buffer in Fig. 4. For each of these values, the similarity $S(u, v)$ to H_k is calculated simultaneously. Then, using a tree adder with argmax calculation, the module determines (u, v) that maximizes S, and the sum and maximum of S. Finally, (u, v) is output as well as the reliability R which is calculated using Eq. (21).

4.4 Block-Mean Optical Flow Calculation Module

From the input optical flow $F = (u, v)$ and its reliability R, the block-mean optical flow $\overline{F}_i = (\overline{u}_i, \overline{v}_i)$ is calculated based on Eq. (22). This module has 3 accumulate registers for each column of block array. We set the block size B to 16 for 480p and 32 for 720p resolution, so there are 40 sets of registers for both resolutions. Using u, v, and R, the module calculates $u \times R$, $v \times R$, and R and adds them to the register set of the corresponding block. Once all the elements of the block i have been accumulated, the values in the register set are put into a queue, and the registers are initialized to 0.

All the processes after this queue are separated from the pipeline explained so far, and controlled by a state machine. A dotted line in Fig. 4 shows this boundary. A set of values in the queue is read one by one, and the block-mean optical flow is calculated using a divider. Since the queue receives new data every $B^2 = 256$ or 1024 clock cycles on average, the divider can process it in an iterative way and does not have to

Table 1 Configurations of parameters

Pattern	f_{BG} [Hz]	A_{BG} [px]	d [px]	f_{FG} [Hz]	A_{FG} [px]	Direction BG/FG	Q [bit]
1	0.4	0	8	8.25	0	V/V	1
2	0.8	5	16	9.00	1	V/D	2
3	**1.2**	10	24	**9.75**	2	D/V	**3**
4	1.6	**15**	**32**	10.50	**3**	**D/D**	-
5	2.0	20	40	11.25	4	-	-

be pipelined. The quotient is represented in the fixed-point format with 8 fractional bits.

4.5 BMFLC module

The BMFLC module is controlled by a state machine and performs the calculation of the band-limited mean optical flow $\overline{F}_i' = (\overline{u}_i', \overline{v}_i')$ and the block weight W_i, as well as the update of the adaptive weight vectors $\boldsymbol{w}_{x,i}$, $\boldsymbol{w}_{y,i}$, based on fixed-point arithmetic with the fractional bit width of 12.

$\overline{F}_i'$ and W_i are calculated in the vertical blanking region, after updating the adaptive weight vectors of all the blocks. First, the reference input vector $\boldsymbol{x}$ is updated using a lookup-table-based sine function module. Then, the adaptive weight vectors of each block are read from memory in turn, which are used to calculate $\overline{F}_i'$ based on Eq. (23) sequentially. W_i is calculated at the same time using Eq. (26) with the threshold $\tau = 1.6$. The results are output to the following module, and $\overline{F}_i'$ is also saved into memory for the next frame.

The update of the adaptive weight vectors is done by calculating Eq. (24) and (25) using the block-mean optical flow $\overline{F}_i$ input from the preceding module and $\overline{F}_i'$ which has already been stored into memory.

4.6 Vibration Component Calculation Module

This module calculates and outputs the extracted vibration component $(\Delta x, \Delta y)$ from the band-limited mean optical flow $\overline{F}_i'$ and the block weight W_i using Eq. (27). This can be done simply by accumulating $\overline{u}_i' \times W_i$, $\overline{v}_i' \times W_i$, W_i using 3 registers and then performing division. Then, the extracted vibration component is sent to an external actuator system, which generates anti-phase vibration to cancel tremor.

5. Evaluation

In this section, we show the evaluation results of the proposed system in terms of extraction accuracy, resource utilization, and latency.

5.1 Accuracy Evaluation Using Synthesized Video

To evaluate extraction accuracy, we synthesized 480p 60-fps video clips with 1200 frames, as shown in Fig. 6, composed of a public-domain image of the retina (background) and a rod-shaped object mimicking a surgical tool (foreground). The background and foreground keep moving back-and-forth. For sub-pixel accuracy, the video is generated from

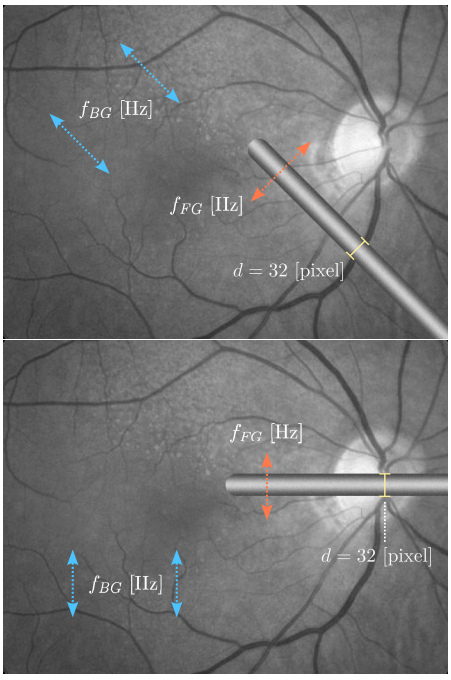


Fig. 6 Synthesized video for evaluation (diagonal/vertical)

8x-oversampled images (5120×3840).

We prepared 29 different video clips with different configurations of the following 6 parameters: frequency and y-axis amplitude of background (f_{BG}, A_{BG}), diameter, frequency, and y-axis amplitude of foreground (d, f_{FG}, A_{FG}), and a combination of background and foreground moving directions. The values used for each parameter is listed in Table 1. For the moving direction, either vertical (V) or diagonal (D) is used, as illustrated in Fig. 6. In the diagonal mode, the y-axis amplitude is also applied to x-axis, so the actual amplitude is $\sqrt{2}A$. To restrict the number of combinations, when we focus on one of the parameters, the other parameters are set to default, which is written in boldface.

Each video is processed using a simulator written in Python that reproduces the exact result of the system implemented on an FPGA. For the default configuration, 3 quantization bit widths ($Q = 1, 2, 3$) are used to evaluate the effect of quantization. A result of the previous system is also evaluated here. Then, the output vibration component is compared with the ground truth based on the mean absolute error between them. We only use Δy here since the algorithm for each axis is substantially the same.

Let Δy at frame k and its corresponding ground truth be $x(k)$ and $y(k)$. To evaluate relative difference, we use the weighted mean absolute percentage error (WMAPE). This is the mean absolute error combined with normalization using the sum of $|y(k)|$, and given as follows:

$$e_{norm} = \frac{\sum_k |x(k) - y(k)|}{\sum_k |y(k)|} \tag{30}$$

It takes 0 only when the extraction result is perfectly accurate and takes 1 when $x(k)$ is always 0. Note that $\sum_k |y(k)|$ is

Table 2 WMAPE (e_{norm}) of each configuration

	Proposed system							Previous System					
Pattern	f_{BG}	A_{BG}	d	f_{FG}	A_{FG}	Dir.	Q	f_{BG}	A_{BG}	d	f_{FG}	A_{FG}	Dir.
1	0.2817	0.2553	0.6084	0.2829	(0/0)	0.2200	0.3615	0.6238	0.6193	0.9325	0.7973	(Inf)	0.8762
2	0.2858	0.2804	0.4802	0.2923	0.2686	0.2867	0.3249	0.6737	0.6208	0.7686	0.7801	0.8132	0.6629
3	**0.3071**	0.2861	0.3490	**0.3071**	0.2593	0.2594	**0.3071**	**0.7908**	0.6646	0.7830	**0.7908**	0.8080	0.9309
4	0.3090	**0.3071**	**0.3071**	0.3447	**0.3071**	**0.3071**	-	0.8647	**0.7908**	**0.7908**	0.7776	**0.7908**	**0.7908**
5	0.3218	0.3266	0.3220	0.3415	0.3700	-	-	0.8889	0.8546	0.7786	0.7783	0.7818	-

Table 3 WMAPE with scale matching (e_{scale}) of each configuration

	Proposed system							Previous System					
Pattern	f_{BG}	A_{BG}	d	f_{FG}	A_{FG}	Dir.	Q	f_{BG}	A_{BG}	d	f_{FG}	A_{FG}	Dir.
1	0.2014	0.2043	0.2659	0.2040	-	0.2096	0.2062	0.2424	0.2647	1.1258	0.4450	-	0.7713
2	0.2040	0.2021	0.2084	0.2094	0.2654	0.1996	0.2015	0.2816	0.2432	0.6665	0.4364	0.9139	0.2663
3	**0.1981**	0.2019	0.2033	**0.1981**	0.2115	0.2025	**0.1981**	**0.5449**	0.2677	0.5877	**0.5449**	0.7538	0.9504
4	0.2032	**0.1981**	**0.1981**	0.2005	**0.1981**	**0.1981**	-	0.6792	**0.5449**	**0.5449**	0.4185	**0.5449**	**0.5449**
5	0.2008	0.1974	0.1995	0.2066	0.1956	-	-	0.7163	0.7417	0.4552	0.3517	0.3732	

almost proportional to $f_{FG} \times A_{FG}$ and is not affected by the other parameters.

As a supplementary criterion, we also use the WMAPE with scale matching, that is, $x(k)$ and $y(k)$ are normalized independently to match scales of both before taking the sum of absolute error between them, as follows:

$$e_{scale} = \sum_k \left| \frac{x(k)}{\sum_k |x(k)|} - \frac{y(k)}{\sum_k |y(k)|} \right| \tag{31}$$

If e_{norm} is high while e_{scale} is low, $x(k)$ and $y(k)$ have different scales but are similar to each other, indicating that it may be possible to reduce error by applying an appropriate scaling factor to $x(k)$. The result of each configuration is summarized in Table 2 and Table 3.

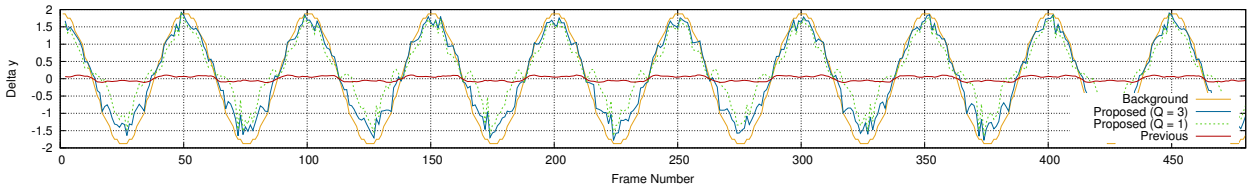
Let us focus on the quantization bit width Q first (all the video parameters are default). Fig. 7 (a) shows the true background motion (displacement from the previous frame) and the average of all the raw optical flow vectors $F(x, y)$ in a frame. Since the background occupies the most area of a frame, the raw average of the proposed system is close to the background motion except for some modulation by the foreground motion. With $Q = 1$, the amplitude tends to be slightly smaller. The raw average of the previous system is very small since optical flow was estimated to be 0 in most background area.

Fig. 7 (b) shows the true foreground motion, which should be extracted, and the extracted vibration component. Though amplitude is smaller in the first few seconds because it takes some time for BMFLC to adapt to the signal, the results of the proposed system show high correlation with the ground truth, since exclusion of the blocks with small block weights, explained in Sect. 3.5 and 3.6, filtered out the optical flow in the background. In addition, the result with $Q = 1$ is close to that with $Q = 3$. As shown in Table 2 and Table 3, the $Q = 1$ configuration brings somewhat higher $e_{norm}(0.3615)$ but has little impact on $e_{scale}(0.2062)$. On the other hand, it is clear that the previous system is severely affected by the background motion, resulting in the higher error values ($e_{norm} = 0.7908$, $e_{scale} = 0.5449$).

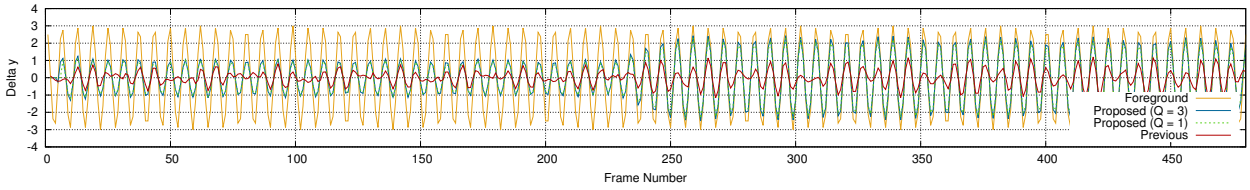
Next, we evaluate how each of the video parameters affects the results. To begin with, increasing f_{BG} and A_{BG}, the frequency and amplitude of the background, causes gradual increase in e_{norm}. This is because blocks on a periphery of the tool are affected by the background motion, interfering with the block-mean optical flow $\overline{F}_i$ (Sect. 3.4) of them. Nonetheless, e_{scale} remains almost constant, and the extracted waveform of $A_{BG} = 0$ and 20 are close to each other as shown in Fig. 8 (a). On the other hand, the previous system suffers from obvious deterioration in accuracy with higher frequency or amplitude, since the filtering mechanism explained in Sect. 2.2 does not work well with large background motion.

Focusing on the tool (foreground) parameters, decreasing the diameter d negatively affects the accuracy since blocks on the boundary become dominant. Especially with $d = 8$, half of the block size $B = 16$, not only e_{norm} but also e_{scale} get high. However, as shown in Fig. 8 (b), the proposed system with $d = 8$ is still superior to the previous system with $d = 40$. If the size of the tool and its distance from a microscope are known in advance, it is possible that scaling with a pre-defined factor helps reduce e_{norm}. Next, increasing the frequency f_{FG} causes error increase to the proposed system partly because of the penalty term in the feature matching defined in Eq. (18). The amplitude A_{FG} is similar, except for the $A_{FG} = 1$ configuration. Possible causes include estimation error of optical flow and quantization error of the ground truth data, which has 1/8-pixel precision. In Fig. 8 (c), it can be confirmed that the waveform of $A_{FG} = 1$ (with 4x scaling) is less stable than $A_{FG} = 4$. Still, it fits the ground truth fairly better than the previous system, which is completely off-track when $A_{FG} = 1$.

Finally, we evaluate the combination of moving directions. In terms of e_{norm}, vertical motion is better than diagonal. With vertical background motion, there is no x-axis motion and thus the weight of blocks on the boundary is kept low. Vertical tool motion allows the tool area to fit the grid of blocks since the diameter $d = 32$ is twice the size of block $B = 16$, which is the ideal situation for the algorithm. How-

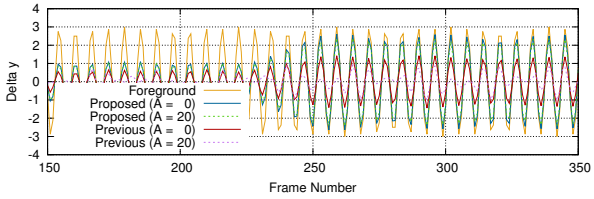


(a) True background motion and average of raw optical flow

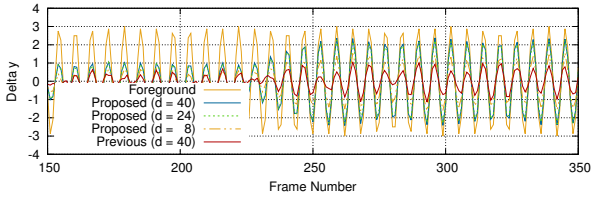


(b) True foreground motion and extracted vibration component

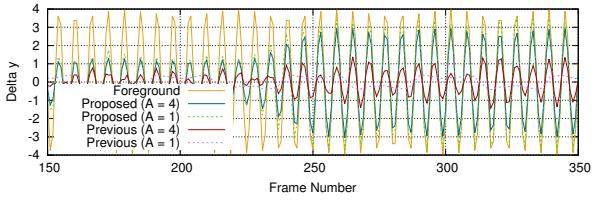
Fig. 7 Processing results for the synthesized video of the default configuration



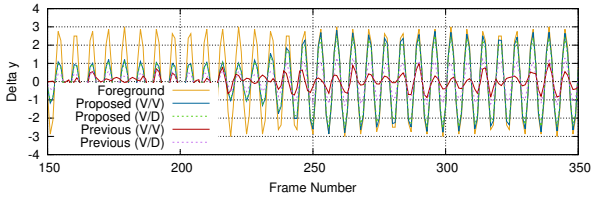
(a) Amplitude of the background A_{BG}



(b) Diameter of the tool d



(c) Amplitude of the tool A_{FG} (with 4x scaling for $A_{FG} = 1$)



(d) Moving direction

Fig. 8 Processing results with different configurations

ever, e_{scale} values and the waveform in Fig. 8 (d) indicate that the result is similar regardless of the combination. In the previous system, on the other hand, extraction accuracy is greatly affected.

Considering the results, the proposed system is able to extract the desired vibration components from video clips with various parameter configurations with consistently higher accuracy than the previous system.

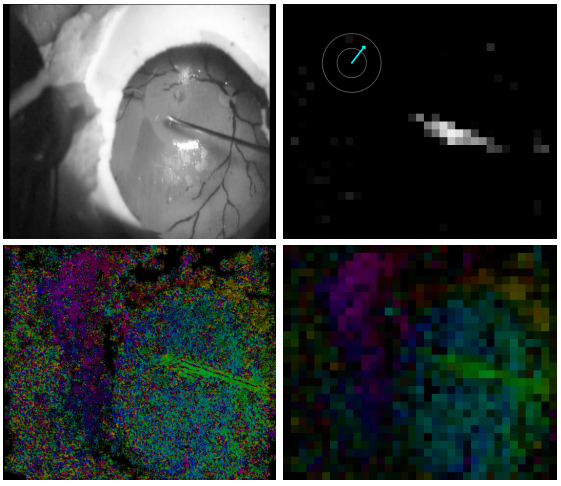


Fig. 9 Processing results using actual video

5.2 Qualitative Evaluation Using Actual Video

As an additional experiment to evaluate the practicability of the system, we processed an actual video clip (560×480) of simulated ophthalmology surgery (epiretinal membrane peel) with the quantization bit width $Q = 3$. In this video, an ophthalmologist is reproducing physiological tremor which is so severe that it is difficult to continue the operation. We observed that the typical amount of movement between two successive frames is not more than 3 pixels at 60 fps. As shown in Sect. 5.1, the proposed system is capable of handling such movement. An example of the visualized result is shown in Fig. 9. The first row shows an input frame and a block weight map with extracted vibration component, and the second row shows raw and block-mean optical flow map, where angle and magnitude are visualized as hue and value in the HSV color space.

Focusing on the block-mean optical flow (bottom-right), we can see that noises in the raw optical flow (bottom-left) are reduced because of the weighted average using the reliability

explained in Sect. 3.3 and 3.4. Based on the block-mean optical flow, the system successfully detects the region of a trembling tool as shown in the block weight map (top-right) as white blocks, though it takes a few seconds for BMFLC to be adapted well. We also qualitatively confirmed that the system can extract the vibration component of the tool with the correct direction, which is shown as the cyan arrow overlaid on the block weight map.

5.3 Resource Utilization

The system was implemented on a Xilinx Zynq-7000 XC7Z020-1CLG400C FPGA, using Vivado 2021.2. Processing system (PS) is not used. The resource utilization is summarized as Table 4. The system configurations are the combinations of 2 resolutions (480p with block size $B = 16$ and 720p with $B = 32$) and 3 quantization bit widths ($Q \in \{1, 2, 3\}$). Note that the result of 720p with $Q = 3$ is omitted since this configuration causes overutilization of BRAM (141) and cannot be implemented. The "OF" columns show the utilization of the modules from gradient calculation (Sect. 4.1) to optical flow estimation (Sect. 4.3), and the "Total" columns show the total utilization including the rest. For reference, the utilization of our previous system [11] for 480p input, which is implemented on a different FPGA (Kintex-7 XC7K325T), is also listed.

According to the table, most of LUTs and FFs are used for optical flow estimation, where a large amount of computations for feature calculation and matching are performed parallelly in a pipeline. Nonetheless, their utilization ratios are well below 40 %. The ratio of DSP for multiplication is even lower. The most dominant resource is Block RAM, which mainly is used as: (a) a frame buffer for the quantized gradient angle θ, (b) line buffers of feature vectors H_{k-1} for feature matching, and (c) memory of the adaptive weight vectors $\boldsymbol{w}$ of BMFLC. The BRAM utilization of OF corresponds to (a) and (b), which is mainly affected by the quantization bit width Q and the image size, as well as the cell size C and the search window size K. Focusing on Q, while (a) changes almost proportionally, the change in (b) explained in Sect. 3.2 is rather gradual. This makes (b) more dominant with smaller Q. On the other hand, (c) mainly depends on the number of blocks. Since 720p resolution with $B = 32$ requires less blocks ($40 \times 22 = 880$) than the 480p resolution with $B = 16$ ($40 \times 30 = 1280$), the BRAM utilization outside OF is less with 720p than 480p.

While BRAM is the most dominant resource, the utilization at 480p is reduced by up to 60 % ($Q = 1$) compared to the previous system which uses an 8-bit frame buffer. This enables the implementation for the smaller FPGA, even with 720p resolution which has 3 times as many pixels as 480p.

5.4 Latency

As shown in Table 4, the maximum operating frequency (F_{max}) is around 123 MHz independently of the configuration. Given that the image size including blanking region is 800×525 for 480p and 1650×750 for 720p, the pixel clock frequency at 60 fps is 25.2 MHz for 480p and 74.25 MHz for 720p, both of which are well lower than the maximum operating frequency. We also confirmed that the implemented system connected with a Digilent Pcam 5C camera works flawlessly at 720p.

We summarize the cumulative latency (clock cycles) from the input of the last pixel of the last block ($I(639, 479)$ for 480p and $I(1279, 703)$ for 720p) to the end of each processing step, in Table 5. Given that a frequency of tremor is 10 Hz, the latency must be less than about 1590 μs to reduce its amplitude by 90 % using anti-phase vibration. Until the adaptive weight vector update, 720p takes about twice as many clock cycles as 480p, which is almost proportional to the image width. On the other hand, latency of the vibration component calculation, which is the most dominant part, depends on the number of blocks. As a result, it takes 49790 cycles at 480p and 43615 cycles at 720p in total to output the estimated vibration component of the next frame.

The previous system, on the other hand, requires 1694 cycles to finish optical flow estimation of each pixel, and 1821 cycles in total to output the extracted result after the last pixel of the frame (480p) is input. This is shorter than the current system since the previous system had narrow search area for optical flow and applied BMFLC filtering only to the mean optical flow of the entire frame. However, considering that it takes 36160 cycles at 480p and 49870 cycles at 720p for the next frame to begin, the estimated vibration component of each frame can be obtained 13630 cycles (540.9 μs) after the frame begins at 480p and before the frame begins at 720p. Therefore, the substantial latency for each resolution is well lower than the threshold for 90 % tremor canceling (1590 μs). It is possible to reduce the latency further by increasing the parallelism of the corresponding modules at the cost of resource utilization, or by using an independent faster clock to drive them. Since the maximum operating frequency is higher than the pixel clock, using a camera with higher frame rate would also help.

6. Conclusion and Future Work

In this paper, we proposed a real-time vibration extraction system using dense optical flow and block-based BMFLC filtering, and implemented the system on an FPGA. The optical flow calculation is based on simple feature matching using the simplified rotation-invariant histogram of oriented gradients. The use of the quantized gradient angle reduced memory resources used for a frame buffer. In addition, the block-based BMFLC filtering enables the detection of blocks belonging to the vibrating objects at low cost. Evaluation results revealed that the proposed system brings consistently better accuracy compared to our previous system.

For future work, we aim to improve the accuracy by introducing better algorithms including new metrics for feature similarity, optical flow reliability, and block weight. Further reduction in Block RAM utilization will help to expand the search window size and support higher resolutions. It is also

Table 4 Resource utilization

	480p ($B = 16$)						720p ($B = 32$)					
	$Q = 1$		$Q = 2$		$Q = 3$		$Q = 1$		$Q = 2$			
Resource	OF	Total	OF	Total	OF	Total	OF	Total	OF	Total	Prev.	Available
LUT	12744	17925	13827	18997	14456	19627	12820	17403	14568	19123	24247	53200
FF	25595	28733	28680	31819	31696	34831	25653	28768	29137	32235	28279	106400
BRAM	21.5	68.5	33.0	80.0	46.0	93.0	51.5	83.5	78.0	110.0	172.0	140.0
DSP	1	7	1	7	1	7	1	7	1	7	54	220
F_{max} (MHz)		122.2		123.9		123.2		124.3		122.4	99.5	-

Table 5 Cumulative latency (clock cycles) at the end of each step

Step	480p	720p
Gradient Calculation	1611	3311
Feature Calculation	5626	11576
Optical Flow Estimation	5639	11589
Block-Mean OF Calculation	5652	11621
Adaptive Weight Vector Update	6549	11894
Vibration Component Calculation	49790	43615

important to perform demonstration experiments in cooperation with an actuator to show the practicability of the system to tremor suppression.

References

[1] C. N. Riviere, R. S. Rader, and N. V. Thakor, "Adaptive Cancelling of Physiological Tremor for Improved Precision in Microsurgery," IEEE Transactions on Biomedical Engineering, vol.45, no.7, pp.839–846, 1998.

[2] K. C. Veluvolu, U. X. Tan, W. T. Latt, C. Y. Shee, and W. T. Ang, "Bandlimited Multiple Fourier Linear Combiner for Real-time Tremor Compensation," Proc. Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS), pp.2847–2850, 2007.

[3] E. Rocon, J. M. Belda-Lois, A. F. Ruiz, M. Manto, J. C. Moreno, and J. L. Pons, "Design and Validation of a Rehabilitation Robotic Exoskeleton for Tremor Assessment and Suppression," IEEE Transactions on Neural Systems and Rehabilitation Engineering, vol.15, no.3, pp.367–378, 2007.

[4] A. As'arry, M. Z. Md zain, M. Mailah, and M. Hussein, "Suppression of Hand Tremor Model Using Active Force Control with Particle Swarm Optimization and Differential Evolution," International Journal of Innovative Computing, Information and Control (IJICIC), vol.9, no.9, pp.3759–3777, 2013.

[5] K. Sajith, V. Darade, and S. Chaudhuri, "Hand Tremor Analysis Using Rigid Body Manipulation in a Dynamic Virtual Haptic Environment," Proc. Conference on Advances In Robotics (AIR), pp.1–5, 2013.

[6] D. Case, B. Taheri, and E. Richer, "Design and Characterization of a Small-Scale Magnetorheological Damper for Tremor Suppression," IEEE/ASME Transactions on Mechatronics, vol.18, no.1, pp.96–103, 2013.

[7] Z. Uhrikova, O. Sprdlik, V. Hlavac, and E. Ruzicka, "Action Tremor Analysis from Ordinary Video Sequence," Proc. Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS), pp.6123–6126, 2009.

[8] K. Cuppens, B. Vanrumste, B. Ceulemans, L. Lagae, and S. Van Huffel, "Detection of Epileptic Seizures Using Video Data," Proc. International Conference on Intelligent Environments (IE), pp.372–373, 2010.

[9] B. Soran, J. N. Hwang, S. I. Lee, and L. Shapiro, "Tremor Detection Using Motion Filtering and SVM," Proc. International Conference on Pattern Recognition (ICPR), pp.178–181, 2012.

[10] X. Wang, S. Garg, S. N. Tran, Q. Bai, and J. Alty, "Hand Tremor Detection in Videos with Cluttered Background using Neural Network Based Approaches," Health Information Science and Systems, vol.9(30), 2021.

[11] T. Manabe, K. Uetsuhara, A. Tahara, and Y. Shibata, "FPGA Implementation and Evaluation of a Real-Time Image-Based Vibration Detection System with Adaptive Filtering," IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol.E103.A, no.12, pp.1472–1480, 2020.

[12] B. D. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision," Proc. International Joint Conference on Artificial Intelligence (IJCAI), pp.674–679, 1981.

[13] B. K. P. Horn and B. G. Schunck, "Determining Optical Flow," Artificial Intelligence, vol.17, pp.185–203, 1981.

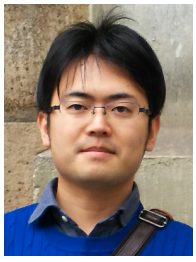
[14] R. J. Elble and W. C. Koller, Tremor, Johns Hopkins University Press, 1990.

[15] B. Widrow and S. D. Stearns, Adaptive Signal Processing, Prentice Hall, 1985.

[16] Z. Luo, J. Chen, T. Takiguchi, and Y. Ariki, "Rotation-Invariant Histograms of Oriented Gradients for Local Patch Robust Representation," Proc. Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA), pp.196–199, 2015.

[17] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.886–893, 2005.

[18] M. J. Swain and D. H. Ballard, "Color Indexing," International Journal of Computer Vision, vol.7, pp.11–32, 1991.



Taito MANABE Taito Manabe received the B.E, M.E, and Ph.D. degrees from Nagasaki University, Japan, in 2016, 2018, and 2021, respectively. Now he is an assistant professor at School of Information and Data Sciences, Nagasaki University. His research interests include real-time processing with an FPGA.

Yuichiro SHIBATA Yuichiro Shibata received the B.E. degree in electrical engineering, the M.E. and Ph.D. degrees in computer science from Keio University, Japan, in 1996, 1998 and 2001, respectively. Currently, he is a professor at Graduate School of Engineering, Nagasaki University. He was a Visiting Scholar at University of South Carolina in 2006. His research interests include reconfigurable systems and parallel processing.