A Study on HFR-Video-Based Software Sensor for Dynamic Scene Analysis

by

Wang Feiyue

Graduate School of Advanced Science and Engineering
Hiroshima University
September, 2023

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Dynamic characteristics such as velocity, displacement and strain can effectively indicate working conditions of machines in factories [1–4] and stability of engineering structures such as bridges and buildings [5–8]. Dynamic analysis plays an important role in structure health monitoring [9] [10], fault detection [11] [12] and service life prediction that can greatly reduce maintenance costs and improve production efficiency.

Effective dynamic analysis requires high-accuracy, high-robustness and high-speed signal capturing and processing. High accuracy guarantees accurate analysis results for further judgment. High robustness provided stable output regardless of changing working conditions. High speed enables long-term and real-time state monitoring that is important to predict and avoid accidents in time. However, limited by complicated structures and different working environments of measured objects, it still keeps challenging to simultaneously meet above mentioned three requirements with an unified method.

Current methods for dynamic measurement can be mainly divided into two categories by their installation requirements: contact-sensor-based and non-contact-sensor-based approaches. Contact-sensor-based methods, such as strain gauges, velometers, and accelerometers, are directly installed on objects to be observed. Strain gauges [13] [14] installed on objects capture strain signals caused by dynamic moving, which can then be easily converted into displacements. Velometers [15] output velocity signals that are linearly proportional to the velocities of target objects, based on the principle of electromagnetic induction. Accelerometers [16–18] are the most popular sensors for contact sensor-based measurements. Acceleration values become more sensitive in higher frequency ranges, and accelerometers are suitable for monitoring the vibrations of fast-vibrating objects with high precision and sensitivity. In recent years, microelectromechanical system accelerometers [19–21] have become popular for structural health monitoring. Most of these accelerometers are inexpensive and designed for real-time long-distance operations that utilize wireless sensor networks. These contact-type sensors can provide accurate and robust dynamic signals; however, their installation is generally time-consuming, and their maintenance costs are high.

Non-contact-sensor-based methods are mainly based on optical sensors. Laser Doppler vibrometers [22–24] can obtain vibration signals by calculating the Doppler shift of a laser beam reflected from a target vibrating object. They are highly accurate and sensitive for repeatable measurements in the frequency range of 0–300 kHz. However, they are easily affected by speckle noise and involve cumbersome intermittent measurements. Vision-based solutions are popular for vibration measurements because of their easy installation and non-contact monitoring. Depth cameras, including infrared cameras [25] [26] and binocular cameras [27] [28] directly measure the actual distances between cameras and vibrating objects and convert their time-changing distance into displacement amplitudes. Typical cameras do not rely on distance calculation; instead, image displacements between adjacent frames are computed using various computer vision algorithms, such as template matching [29] [30] , optical flow [31] [32] , and digital image correlation [33] [34]. Video-based solutions require considerable memory for data storage and time-intensive data processing. Moreover, their frame rates are often limited to tens of frames per second, and most of them provide dynamic measurement in a low frequency range within tens of hertz.

To realize higher-frequency-range measurement, high-speed vision systems have been developed to measure dynamic signals in a high frequency range. High-speed vision systems [35–37] have been developed to measure dynamic signals in a high frequency range. Compared with standard video formats at low frame rates, high-speed vision systems can execute video processing for HFR images at a high frame rate of hundreds or thousands of frames per second, enabling dynamic measurements at a frequency of 500 Hz or higher. However, captured HFR images can only be displayed in slow motion to operators on an offline computer display; thus there is a demand for further real-time processing and visualization of high-speed information for more intuitive and widely applications.

1.2 Concept of the research

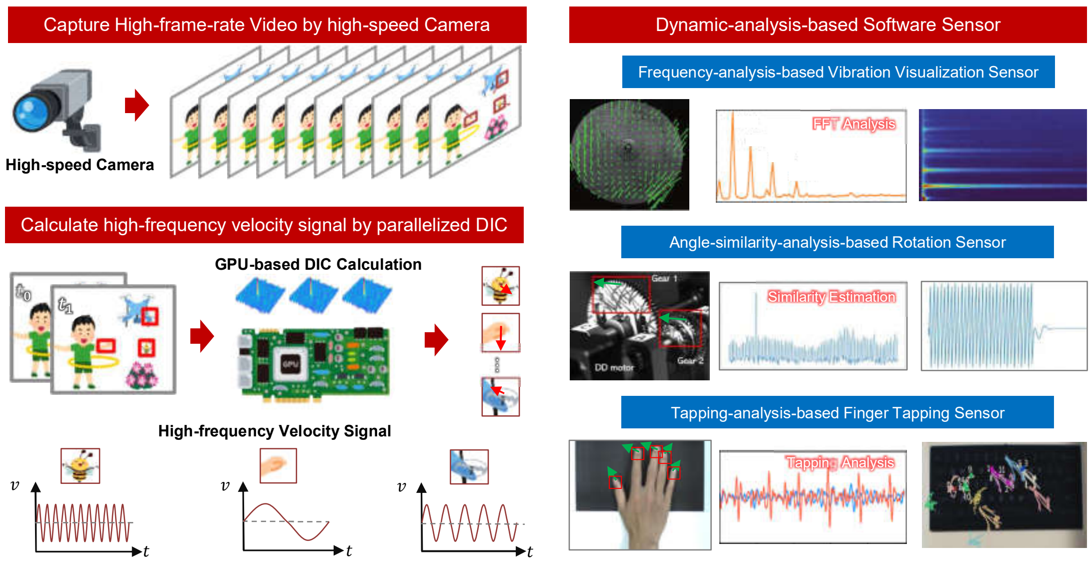

Capture High-frame-rate Video by high-speed Camera
High-speed Camera
Calculate high-frequency velocity signal by parallelized DIC
GPU-based DIC Calculation
High-frequency Velocity Signal
Dynamic-analysis-based Software Sensor
Frequency-analysis-based Vibration Visualization Sensor
FFT Analysis
Angle-similarity-analysis-based Rotation Sensor
Similarity Estimation
Tapping-analysis-based Finger Tapping Sensor
Tapping Analysis

Figure 1.1: Concept overview of this study.

In this study, we propose a novel concept called High-frame-rate(HFR)-video-based software sensor that combines high-speed digital image correlation (DIC) with real-time dynamic analysis to analyze high-speed and micro movements happening at dozens or hundreds of Hz that cannot be directly seen by the human eye.

Figure 1.1 shows the concept of HFR-video-based software sensor that mainly consists of three steps: High-frame-rate (HFR) video capturing, high-speed DIC calculation and real-time dynamic analysis.

High-speed camera working at hundreds of fps can comprehensively sample motion information of fast-moving objects with a high sample frequency. However, huge image data also brings challenges in real-time image processing that requires computer vision algorithms must work within millisecond-level time.

To address this challenge, I develop GPU-based DIC algorithms that parallelize a batch of DIC calculation by the strong GPU platform to estimate velocity values of multiple regions simultaneously in milliseconds. Realtime and frame-by-frame DIC calculation converts the high-speed-vision camera into a high-frequency-velocity camera that can output high-accuracy and high-frequency velocity signals for further dynamic scene analysis.

Based on the high-frequency-velocity camera, we developed three kinds of high-speed-vision-based software sensors by realtime dynamic analysis of the velocity signals, namely, frequency-analysis-based vibration visualization sensor, angle-similarity-analysis-based rotation sensor and tapping-analysis-based finger tapping sensor.

Vibration visualization sensor can estimate and display vibration distributions at all frequencies in real time to help operators intuitively monitor high-speed vibration. The proposed sensor can estimate the full-field vibration displacements of 1920×1080 images in real time at 1000 fps and display their frequency responses in the range of 0–500 Hz on a computer at dozens of frames per second by accelerating phase-only DICs for full-field displacement measurement and video conversion. The effectiveness of this sensor for real-time vibration monitoring and visualization was demonstrated by conducting experiments on objects vibrating at dozens or hundreds of hertz.

Rotation sensor can simultaneously detect the angles of multiple rotational objects in a high-speed video sequence. Our rotation sensor can be executed at 500 fps using parallel-implementing digital image correlation processes to inspect the similarities between the input image and 360 reference images at different angles on a GPU-based high-speed vision system. Its performance in measurement accuracy was verified using several experiments for multiple rotating gears monitored with partial occlusion in 500-fps videos, including multiple gears fast-rotating at 2400 rpm.

Finger tapping sensor can simultaneously estimate when and where an operator taps with his/her finger by detecting the high-frequency component that develops when the fingertip actively contacts something. Our fingertapping sensor can execute DIC operation on 720×540 resolution images at 500 fps with CNN-based fingertip detection at 30 fps. By presenting several experimental results for finger tapping detection, including virtual keyboard interaction with a tenfinger keyboard input, the effectiveness of our fingertip sensor as a finger tapping interface was demonstrated, which can simultaneously estimate the tapping positions and moments of multiple fingers when finger tapping is performed 10 times or more in a second.

1.3 Outline of thesis

This thesis is organized as 7 Chapters, including this introduction.

Chapter 2 summarized related works on digital image correlation and high-speed vision.

In Chapter 3, two DIC algorithms are proposed for high-speed dynamic information sensing based on high-speed vision system. GPU-based Batch POC (Phase-only Correlation) algorithm is designed to simultaneously estimate velocity values of multiple regions within milliseconds by parallelizing phase correlation calculation on the strong GPU platform. To achieve higher accuracy and robustness of velocity estimation, multi-POC algorithm is proposed based on Batch POC. Multi-POC utilizes multiple reference images generated by subpixel intrpolation instead of original single one as reference to get optimal estimation of image velocity. The performance of the two algorithms are also evaluated in this section.

Chapter 4 introduce the concept and algorithms of proposed frequency-analysis-based vibration visualization sensor. A high-speed-vision system is developed to visualize the vibration distribution at hundreds of Hz. Finally, two experiments containing free vibration and forced vibration are presented to demonstrate the effectiveness of the vibration visualization sensor.

In Chapter 5, to measure the angle speed of rotating parts in factories, the high-speed-vision-based rotation sensor is developed. The concept of rotation sensor is firstly introduced. Relative similarity estimation algorithm is designed and run on the developed high-speed-vision-based system. To verify the performance of proposed rotation sensor, offline analysis of multiple rotary gears and real-time experiment for high-speed rotary gears are further conducted.

Chapter 6 aims to explain the research of fingertip velocimeter for multi-finger tapping detection. Requirement of real-time multi-finger tapping detection and limitation of current methods are presented in the introduction section. To address this requirement, high-speed-vision-based fingertio velocimeter is introduced in the concept section and the algorithm section and implemented in the system section. The performance of proposed fingertip velocimeter is verified and compared by the force sensor experiment and the comparison experiment, respectively. Moreover, a kind of virtual keyboard is also developed based on proposed fingertip velcocimeter and presented in the experiment section.

In Chapter 7, it summarized the contributions of this study and discussed future work.

Chapter 2

Related works

2.1 Digital Image Correlation

DIC [38] is a well-known image-based measurement technique that can precisely estimate deformation displacements in images as a full-field distribution by calculating the similarities between digital images before and after deformation. Owing to its easy experimental setup and effective measurement, DIC has been widely applied in the field of experimental solid mechanics to quantitatively analyze the deformation displacements of materials [39], components [40], and structures [41]. DIC was proposed in 1980s to track the motion of a small aluminum specimen [42]. Many studies have been conducted to improve the performance of DIC by focusing on better similarity estimation criteria, such as sum of absolute differences [43], sum of squared difference [44], cross-correlation [45], and zero-mean normalized cross-correlation [46]. To improve the computational efficiency of DIC, Sutton et al. [47] proposed employing the Newton–Raphson (NR) method with differential correlation to accelerate normalized cross correlation, and Chen [48] used a fast Fourier transform red (FFT) to estimate the similarity in the frequency domain and detected their peaks to determine displacements without pixel-by-pixel searching.

To improve the measurement accuracy of DIC, many methods have been proposed for subpixel image registration, such as iterative space-domain cross-correlation [49], gradient-based subpixel registration [50], and genetic algorithms [51]. Pan [46] demonstrated that the iterative NR algorithm achieves higher accuracy and better stability, and proposed the inverse-compositional Gauss–Newton algorithm [52] for more efficient subpixel registration. To improve the subpixel interpolation accuracy, Luu [53] employed B-spline interpolation with a family of recursive interpolation schemes, and many optimized versions of B-spline-based interpolation [54, 55] have been introduced for precise DIC computation. Several studies have aimed to reduce interpolation errors by introducing pre-processing techniques such as the random subset offset strategy [56], self-correlation scheme [57], and Gaussian pre-filtering [58].

However, most DIC algorithms cannot realize real-time visualization of full-field displacements or velocities. This is because they focus on the accuracy of short-term videos, which are not designed for long-term and real-time applications. Heavy computation in DIC remains a challenging limitation in the development of real-time and long-term displacement monitoring. Moreover, most targets such as industrial operating machines, have poor surface texture, making it difficult to attach markers on their surfaces. Furthermore, weak-texture targets, small image sizes [59], and different displacement ranges [60] [61] can decrease DIC performance heavily.

In this study, to address real-time and full-field displacement in high-speed-vision systems, GPU-based batch POC algorithm is developed. Moreover, Batch-POC-based multi POC is further proposed to achieve more accurate displacement estimation under challenging cases.

2.2 High-speed vision

With the rapid development of computer and electronic device technologies, numerous high-speed vision systems have been developed and applied to various high-speed motion applications that cannot be directly observed by the naked eye, such as object tracking [62] [63], biomedicine [64], robotic control [65], high-dynamic-range image capturing [66], and 3D measurement [67]. Wantanabe et al. [68] developed a high-speed vision system that can extract the moment features of 1024 objects in a 256×256 image at 955 fps using a parallel-implemented labelling algorithm on an field programmable gate array (FPGA). To improve the allowed complexities of algorithms implemented on high-speed vision systems, Ishii et al. [62] developed a personal-computer(PC)-based high-speed vision system that can transfer HFR images and their processed image features in real time by organically linking three data-processing platforms: FPGA, CPU, and GPU. For real-time video processing at a high frame rate, various types of computer vision algorithms have been implemented on this PC-based high-speed vision system such as face tracking [69], multiobject feature tracking [70], color-histogram-based tracking [71], and optical flow estimation [72].

Recently, high-speed vision systems have been used for vibration sensing. Considering an image sensor as a set of numerous optical sensors in which every pixel can measure time-varying brightness as a signal for time-series analysis, an HFR camera, in which HFR images can be used to observe human-invisible vibrations at the audio-frequency level, can provide full-field vibration signals sampled at its frame rate. Jiang et al. [73] achieved robust tracking of vibrating objects by executing pixel-level digital filters for HFR images and performed real-time tracking by processing 512×512 images at 1000 fps [74]. Shimasaki et al. [75] reported the pixel-level localization of flying honeybees with wing flapping at 180–240 Hz and estimated their trajectories by calculating the frequency responses of brightness signals at all pixels of 1024×1024 images at 500 fps. Similarly, HFR video-based tracking was conducted for flying multicopters with propellers at dozens of rotations per second by performing pixel-level STFTs in real time at 500 fps [76].

However, captured HFR images can only be displayed in slow motion to operators on an offline computer display; thus there is a demand for further real-time processing and visualization of high-speed information for more intuitive and widely applications.

In this study, three kinds of high-speed-vision-based software sensors are proposed to convert high-speed sensor information into real-time dynamic analysis results that are more intuitive for direct applications.

Chapter 3

GPU-based DIC algorithms for high-speed sensing

3.1 Introduction

Digital image correlation (DIC) is a popular vision-based measurement method that can estimate surface displacement by calculating the similarity between images before and after deformation. Owing to its advantages, including full-field view, no-contact measurement, and easy installation, DIC has been widely applied in many experimental mechanics.

With the development of vision chips, great increase of frame rate and resolutions of current cameras makes it more and more time-consuming to execute full-field displacement estimation by DIC methods. As mentioned in Section 2, many current DIC-based displacement-analysis systems focus on high accuracy of DIC by offline processing which enables complicated algorithm design. However, in high-speed motion case, offline processing requires huge physical memory for storing high-resolution and high-frame-rate image data and only supports a short-time video analysis. Moreover, offline processing cannot provide feedback information in time which is critical for quick response to accidents that can effectively avoid big loss in factories. Thus, for long-term and real-time full-field displacement calculation, higher speed of DIC calculation is meaningful and

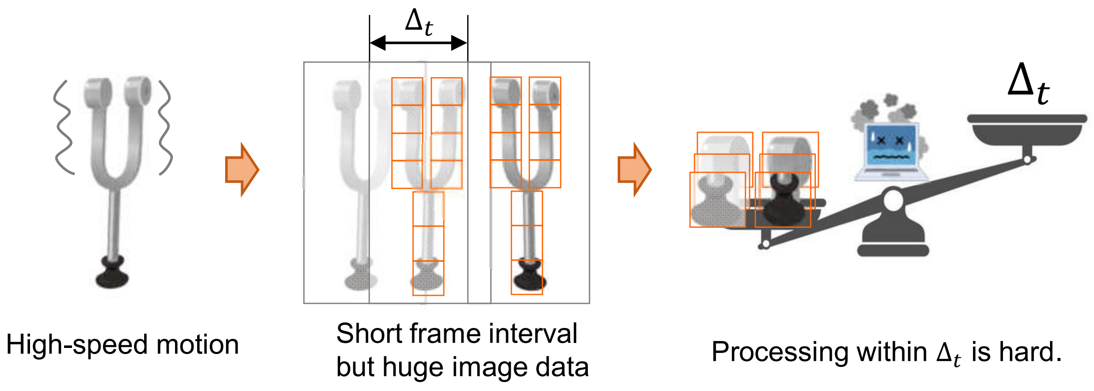

Δ_t
Δ_t
High-speed motion
Short frame interval but huge image data
Processing within Δ_t is hard.

Figure 3.1: Conflict between heavy full-field computation and short frame interval.

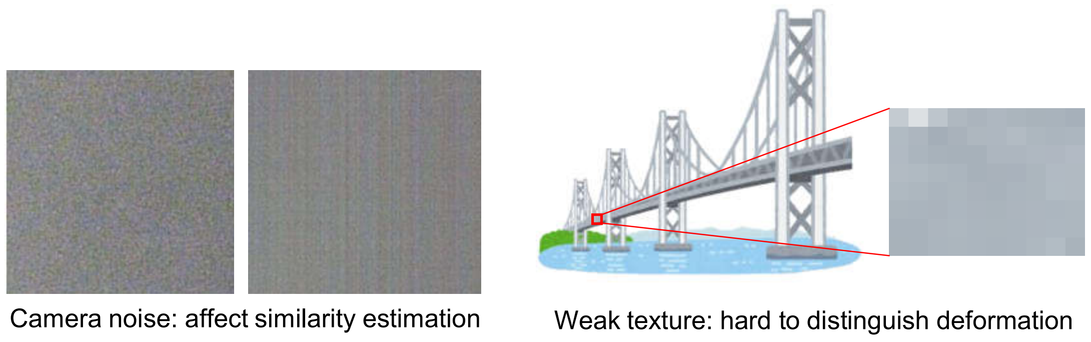

Camera noise: affect similarity estimation
Weak texture: hard to distinguish deformation

Figure 3.2: Noise and weak texture will decrease performance of similarity estimation.

necessary.

Figure 3.1 shows the conflict between slow DIC calculation for full-field measurement and short frame interval of high-speed vision systems. It is challenging to process multiple pairs of subimages within δ_t that is generally several milliseconds in a high-speed vision system working at hundreds of frames per second. One-by-one DIC calculation requires high configuration of computers and only work well under low-resolution images.

Another challenge in DIC calculation is that the measuring environments in real applications are not always conducive. As Figure 3.2 shown, random noise and pattern noise will affect the similarity calculation in DIC procedure. Low signal noise ratio (SNR) usually causes unreliable estimation results. Moreover, weak-texture object cannot provide enough effective information for the similarity comparison of images before and after deformation. Generally, it is also hard to attach markers as artificial texture information in big-scale structures or industrial operating machines.

In this study, to address these two challenges, we proposed two kinds of GPU-based high-speed DIC algorithms, namely, GPU-based Batch POC and GPU-based Multi POC, respectively. Batch POC accelerate DIC calculation of multiple pairs of subimages by parallelizing all the steps of phase-only correlation on the strong GPU platform. Through extensive parallelization, our developed Batch POC can estimate displacements occurring in the whole view of an image of 1920×1080 resolution at a millisecond time scale. Multi POC is developed to improve the estimation accuracy in challenging application cases by calculating the mathematical expectations of multiple independent DIC estimations as the measured true values. Average of estimation results calculated by multiple reference images leads to an unbiased displacement estimation and performs better than single-time estimation on noisy or weak-texture images.

3.2 GPU-based Batch POC

3.2.1 concept

Input images of $M \times N$ pixels are converted to displacement images $\mathbf{A}(i, j, k\tau)$ of $M' \times N'$ pixels:

$$\mathbf{A}(i,j,k\tau) = \mathrm{Displacement}(I(x,y,k\tau)) \; (k = 1,2,\cdots), \tag{3.1}$$

where the $M'N'$ displacement sensors are virtually located on the input images, and they are operated at the same sampling time as the camera cycle time τ. $\mathbf{A}(i, j, k\tau) = (A_x(i, j, k\tau), A_y(i, j, k\tau))$ is composed of x- and y-displacement images with $M' \times N'$ pixels. Considering the computing resources for real-time execution, their resolutions are downconverted from $M \times N$ pixels.

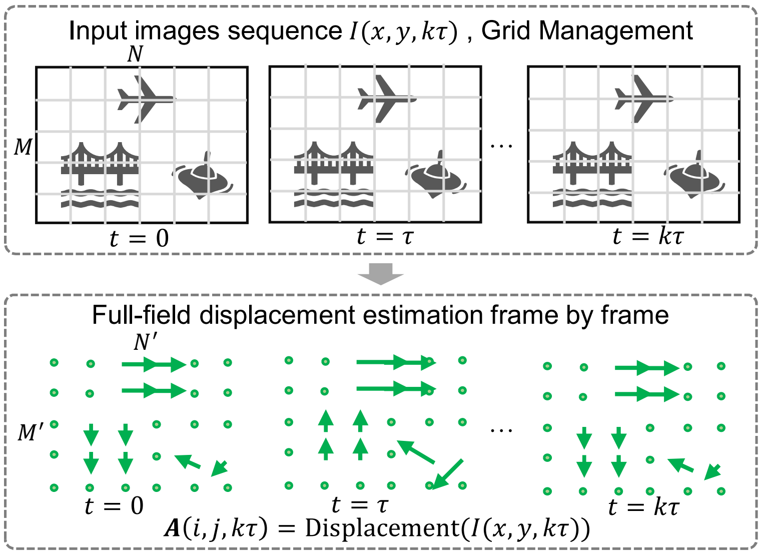

Input images sequence $I(x,y,k\tau)$, Grid Management
N
M
$t=0$
$t=\tau$
$t=k\tau$
Full-field displacement estimation frame by frame
N'
M'
$t=0$
$t=\tau$
$t=k\tau$
$\boldsymbol{A}(i,j,k\tau)=\text{Displacement}(I(x,y,k\tau))$

Figure 3.3: Full-field displacement estimation.

As Fig. 3.3 shown, full-filed displacement estimation consists of two steps, namely, grid management and displacement estimation. Grid management defines the density and positions of sampled regions from the whole image range. Displacement is parallelized DIC operation that can parallelize DIC operation to speed up the displacement estimation of all sampled regions.

3.2.2 Implementation

Parallel Grid Management on GPU: To parallelize the DIC computation for full-field displacement measurement, we assign grid management for creating multiple blocks to be processed in parallel on the GPU-based high-speed vision platform. Given input images $I(x, y, t)$ of $M \times N$ pixels, the four parameters for grid management were block size (b_x, b_y) and the block step (s_x, s_y). They determined the accuracy and density of the full-field displacement measurements. The block and step sizes determined the $M' \times N'$

resolution of the full-field displacement measurement as follows:

$$\begin{cases} N' = \left\lfloor \dfrac{(N - b_x + s_x)}{s_x} \right\rfloor \\ M' = \left\lfloor \dfrac{(M - b_y + s_y)}{s_y} \right\rfloor \end{cases} \tag{3.2}$$

where $\lfloor x \rfloor$ denotes the largest integer that is less than or equal to x.

DIC-based Displacement Measurement: For full-field displacement measurements, we parallelized the phase-only correlation (POC) [77] algorithm, which can utilize efficient peak detection in the frequency domain for subpixel image displacements. The displacements between the two sub-images $I_{ij}(x, y, t)$ and $I_{ij}(x, y, t_R)$ were calculated as follows. In the following steps, t and t_R indicate the times at the current frame and the reference frame in the DIC computation, respectively.

(1) 2-D FFTs of the input and reference sub-images

The sub-images of $b_x \times b_y$ pixels were converted into the frequency domain as follows:

$$G_{ij}(u, v, t) = \mathcal{F}(I_{ij}(x, y, t) \cdot Q(x, y)), \tag{3.3}$$

$$G_{ij}(u, v, t_R) = \mathcal{F}(I_{ij}(x, y, t_R) \cdot Q(x, y)), \tag{3.4}$$

where $\mathcal{F}(\cdot)$ indicates the 2-D FFT function. $Q(x, y)$ is a binary mask image that indicates the pixels to be processed in the DIC computation.

(2) Cross-power spectrum computation

The phase correlation distribution in the frequency domain was computed using the following cross-power spectrum:

$$R_{ij}(u, v, t) = \frac{G_{ij}(u, v, t) \cdot G^*_{ij}(u, v, t_R)}{\left| G_{ij}(u, v, t) \cdot G^*_{ij}(u, v, t_R) \right|}, \tag{3.5}$$

where G^*_{ij} is the complex conjugate of G_{ij}.

(3) Inverse 2-D FFT for peak detection

The cross-power spectrum $R_{ij}(u, v, t)$ was transformed into the spatial domain, as follows:

$$r_{ij}(x, y, t) = \mathcal{F}^{-1}(R_{ij}(u, v, t)), \tag{3.6}$$

where $\mathcal{F}^{-1}(\cdot)$ indicates the inverse 2-D FFT function.

The displacement vector $\Delta\mathbf{x}_{ij}(t) = (\Delta x_{ij}(t), \Delta y_{ij}(t))$ at time t was obtained with integer pixel precision by determining the maximum peak value of $r_{ij}(x, y, t)$, as follows:

$$\Delta\mathbf{x}_{ij}(t) = (\Delta x_{ij}(t), \Delta y_{ij}(t)) = \arg\max_{x,y}\ r_{ij}(x, y, t). \tag{3.7}$$

(4) Estimation of the subpixel displacement

To estimate the displacement vector with higher accuracy, subpixel-level peak detection was conducted by computing the weighted centroid values of the correlation values $r_{ij}(x, y, t)$ in the neighborhood $N(\Delta\mathbf{x}_{ij}(t))$ as follows:

$$\Delta\tilde{x}_{ij}(t) = \frac{\sum\limits_{(x,y)\in N(\Delta\mathbf{x}_{ij}(t))} x \cdot r_{ij}(x, y, t)}{\sum\limits_{(x,y)\in N(\Delta\mathbf{x}_{ij}(t))} r_{ij}(x, y, t)}, \tag{3.8}$$

$$\Delta\tilde{y}_{ij}(t) = \frac{\sum\limits_{(x,y)\in N(\Delta\mathbf{x}_{ij}(t))} y \cdot r_{ij}(x, y, t)}{\sum\limits_{(x,y)\in N(\Delta\mathbf{x}_{ij}(t))} r_{ij}(x, y, t)}. \tag{3.9}$$

where the displacement vector $\Delta\tilde{\mathbf{x}}_{ij}(t) = (\Delta\tilde{x}_{ij}(t), \Delta\tilde{y}_{ij}(t))$ corresponds to the displacement image of $M' \times N'$ pixels in the DIC-based displacement measurement.The neighborhood $N(\mathbf{x})$ indicates the $P \times P$-pixel neighborhood of $\mathbf{x}$, where P is an odd positive integer.

Table 3.1: Execution time of DIC-based displacement measurement (unit: ms)

block size / block step	64 / 64	64 / 32	128 / 128	128 / 64	128 / 32	256 / 256	256 / 128	256 / 64
Copy from CPU to GPU	0.180	0.180	0.180	0.180	0.180	0.180	0.180	0.180
(1) 2-D FFT	0.232	0.555	0.361	1.270	4.446	0.441	1.848	6.380
(2) Cross Power Spectrum	0.060	0.226	0.059	0.207	0.793	0.056	0.187	0.650
(3) Inverse 2-D FFT	0.199	0.590	0.304	0.957	3.235	0.457	2.026	6.392
(4) Subpixel Estimation	0.009	0.010	0.008	0.009	0.009	0.008	0.008	0.008
Copy from GPU to CPU	0.040	0.038	0.035	0.054	0.062	0.061	0.051	0.069
Total Time	**0.720**	**1.599**	**0.947**	**2.677**	**8.725**	**1.203**	**4.300**	**13.679**

$A_d(i, j, t)$ to be processed in the STFT-based vibration visualization was set to the x- or y-displacement component based on the vibration characteristics:

$$A_d(i, j, t) = \Delta\tilde{x}_{ij}(t) \ \text{ or } \ \Delta\tilde{y}_{ij}(t). \tag{3.10}$$

Subprocesses (1)–(4) were simultaneously executed for $M'N'$ blocks in parallel on a GPU at time $t = k\tau$ every time an input image of $M \times N$ pixels was captured.

3.2.3 Execution time

We accelerated the algorithm by implementing it on a GPU board (GeForce RTX 3090) with the C++ language and CUDA Toolkit 11.4 using Microsoft Visual Studio Community 2017.

Table 3.1 lists the execution time of the DIC-based displacement measurement for 8-bit 1920×1080 images ($M = 1920$, $N = 1080$) when the block size b ($= b_x = b_y$) and step s ($= s_x = s_y$) were set to $b = 64$, 128, and 256, and $s = b$, $b/2$, and $b/4$, respectively. The $M' \times N'$ resolution in the full-field displacement measurement was determined using Eq.(3.2). The DIC computation was accelerated by parallelizing it to use the global memory on the GPU for $M' \times N'$ threads, corresponding to the block operation of $b \times b$ pixels. The image and processed data transfer times between the CPU and GPU are not negligible, and the execution time listed in Table 3.1 involved the time required to transfer (a) two 8-bit-unsigned-char input images of 1920×1080 pixels from the CPU to the GPU and (b) the full-field 32-bit-float displacement vectors of $M' \times N'$ blocks from the GPU to the CPU. The execution time of the DIC computation was similar when the block size was equal to the block step. 0.720 ms ($b = s = 64$), 0.947 ms ($b = s = 128$), and 1.203 ms ($b = s = 256$). When the block size was larger than the block step, the execution time increased. 8.725 ms ($s = 32$), 2.677 ms ($s = 64$), and 0.947 ms ($s = 128$), with $b = 128$.

3.3 GPU-based Multi POC

3.3.1 Concept

The proposed MFPOC is illustrated in Fig. 3.4. Different from one-to-one single-time estimation in traditional DIC methods, here, multiple reference images were generated with different subpixel displacements by interpolating the reference image and realizing multi-to-one correlation calculation in parallel on a GPU platform. Final displacement synthesized multiple independent estimations by averaging the values for high statistical robustness. Compared to traditional one-to-one POC, the MFPOC has the following advantages.

(1) More robust against image noise: Random noise decreases similarity between images and yields poor correlation results. The MFPOC calculates the correlation values between the test image and the multiple reference images with different noise distributions. The average operation can considerably reduce the error caused by random noise.

(2) More accurate displacement estimation of small-size images: Small-size images cannot provide sufficient detailed information near the correlation peak for accurate interpolation to calculate subpixel-level-accuracy results. In MFPOC, subpixel estimation

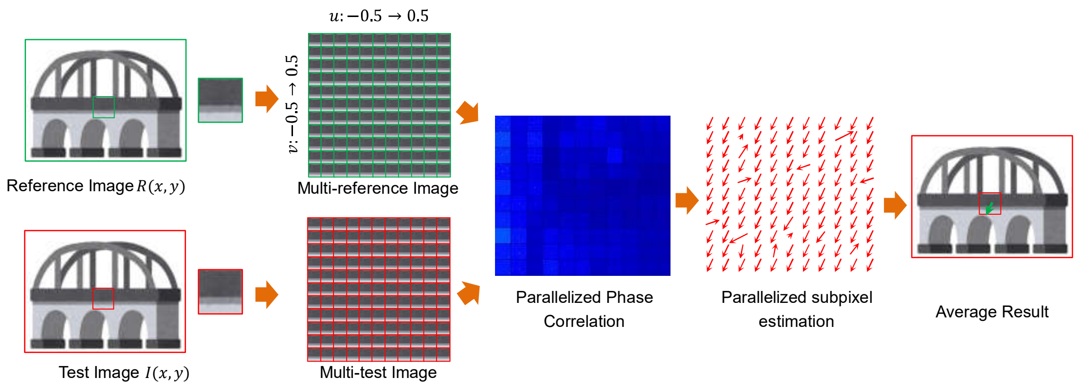


Figure 3.4: Concept of multi POC.

is independently executed on every correlation map. Subpixel estimation error can be considerably suppressed by averaging the results of multiple estimations.

(3) Better adaptability to the displacement range: The subpixel estimation error is small in the range of 0.2 to 0.2 pixels and large above the 0.2-pixel offset [78]. In MFPOC, multiple reference images were generated corresponding to the subpixel displacements from 0.5 to 0.5 pixels overlapping the entire subpixel displacement range. Symmetric rough estimations can cancel each other to a certain extent by performing the averaging operation.

3.3.2 Implementation

Given the reference image $R(x, y)$ and test image $I(x, y)$ of $W \times H$ pixels, the procedure for Multi POC can be divided into following three steps.

(1) L reference image generation

For complete subpixel-level displacements for independent displacement estimation, we generated L reference images corresponding to the subpixel range from 0.5 to 0.5 in both horizontal and vertical directions with a step of $s = \frac{1}{\sqrt{L}-1}$. To generate a reference image $R_l(x, y)$ with a subpixel displacement of $d_l = (\triangle x, \triangle y)$, where $l = 1, 2, \ldots, L$, bilinear interpolation was used to upsample the reference image $I(x, y)$ to a higher-resolution image $I'(x, y)$ of $\frac{W}{s} \times \frac{H}{s}$ pixels.

$$I'(x, y) = \text{Bilinear}(I(x, y), \frac{1}{s}). \tag{3.11}$$

Then, the shifted reference image $I'(x - \frac{\triangle x}{s}, y - \frac{\triangle y}{s})$ with an inter offset of $(\frac{\triangle x}{s}, \frac{\triangle y}{s})$ from $I'(x, y)$ was easily obtained via the shifting operation. Finally, with a down-sampling operation, the final reference image $R_l(x, y)$ with a subpixel displacement of $(\triangle x, \triangle y)$ can be obtained as

$$R_l(x, y) = \text{Downsample}(I'(x - \frac{\triangle x}{s}, y - \frac{\triangle y}{s}), s). \tag{3.12}$$

As the reference image $I(x, y)$ was fixed during the displacement estimation, reference-image generation only required one-time execution in advance. Therefore, no computational burden occurs in real-time displacement estimation.

(2) Parallelized phase-only correlation

Displacement between every generated reference image $R_l(x, y)$ and test image $I(x, y)$ is estimated independently by phase-only correlation.

$$(\triangle x_l, \triangle y_l) = \text{POC}(R_l(x, y), I(x, y)). \tag{3.13}$$

The estimated result should be compensated by the subpixel displacement of the reference image to obtain the correct values.

$$e_l = (\triangle x_l, \triangle y_l) - d_l. \tag{3.14}$$

Multiple reference images with different subpixel displacements can provide different precision estimations owing to different offsets and similarities. However, it is a time-consuming process compared to single-time estimation. To solve this problem, we parallelized the L-time POC operations on the GPU platform, thereby significantly improving the speed of MFPOC.

(3) Averaging the results to obtain an unbiased estimation

L estimated results were averaged to obtain their mathematical expectation value to realize an unbiased estimation.

$$e = \frac{\sum_{i=1}^{L} e_l}{L}. \tag{3.15}$$

The averaging operation can effectively reduce the random noise caused by interpolation or other numerical procedures. Specifically, when the image has poor quality or small size, multiple reference images can provide significantly more redundant information for robust displacement estimation.

3.3.3 Execution time

The performance of the proposed MFPOC algorithm was verified using C++ language on a personal computer (Dell, USA) equipped with a GPU board of NVIDIA Geforce RTX 2080Ti (NVIDIA, Santa Clara, US), an Intel Core CPU i9-9900 K @ 3.60 GHz and 32 GB- memory. The PC had Windows 10 Professional 64-bit OS (Microsoft, Redmond, WA, US). For the MFPOC algorithm parallelized on the PC, reference image count L was set to 121. The subpixel range in the horizontal and vertical directions was from 0.5 to 0.5 with a step of 0.1. Reference images were generated only once in an offline mode on the CPU platform. L independent POC operations were parallelized on the GPU platform.

Execution times of the Multi POC algorithm for different image sizes are presented and compared with the POC function of OpenCV (Release version 4.5.0) in Table 3.2.

For the POC function of OpenCV lib, multiple POC only needs to convert the ref-

Table 3.2: Execution time of Multi POC (unit: ms)

Time:ms	64×64	128×128	256×256
Single POC (OpenCV)	0.301	0.855	3.288
Multiple POCs (OpenCV)	9.252	34.832	153.118
Multi POC	0.223	0.822	3.851

erence image as a float type for FFT operation, because the test image does not need to be converted repeatedly. Thus, the execution time of multiple POC is not simply 121 times that of the single POC. For all image sizes, the execution times of multiple POCs of OpenCV were 30–50 times that of the single POC. For an image size of 64×64 pixels, the MFPOC required 0.223 ms. This is faster than that required for the single POC of OpenCV, which required 0.301 ms. For bigger images (256×256 pixels), multiple POC of OpenCV required 153.118 ms; this cannot meet the real-time requirements. However, the MFPOC maintained almost the same speed for the single POC of OpenCV at a rate of 259 fps.

3.3.4 Experiment

(1) Parameter Experiment

In this section, we demonstrate the performance of the Multi POC algorithm by comparing it with the POC function of OpenCV under the cases of small-size and noisy images that are challenging for traditional single-estimation DIC algorithms. To guarantee the rationality and fairness of the experiments, we collected 12 classic images in the computer vision field: “baboon,” “cameraman,” “Lena,” “peppers,” “house,” “bridge,” “couple,” “airplane,” “dark-hair woman,” “man,” “woman,” and “sailboat,” as shown in 3.5.

Every image was resized to: 64×64, 128×128, and 256×256 pixels. For every image size, 10 images with subpixel displacements in the horizontal direction in the range of 0.0 to 0.9 were generated.

Reference image count L is the most important parameter of Multi POC. Here, we

Figure 3.5: Dataset images.

designed a parametric experiment to discuss the effect of the reference image count on the performance of Multi POC. To this end, we test the averaged estimation errors of different reference image counts $L = (2k + 1)^2$, where $k \in [3, 10]$, and the displacement step s was set to 0.1. The averaged estimation error can be calculated as follows:

$$error = \frac{\sum_{i=0}^{9} \left| e_i(x) - x_T^i \right|}{10}, x_T^i = i \times 0.1. \tag{3.16}$$

Where $e_i(x)$ and x_T^i are the estimated and true values in the horizontal direction, respectively. The experimental results are shown in Fig 3.6.

When k was smaller than 5, the average error decreased with increasing k value. However, when k continued to increase, the performance showed no obvious improvement. However, a bigger k value increases the computational time. Considering the balance of performance and running speed, we set k to 5 in this study, i.e., L= 121.

(2) Noise Experiment

In real applications, various types of image noise are generated by the imaging system. To verify the performance of the MFPOC algorithm in real cases, we added two most common types of noise to the images in our dataset: Poisson noise and Gaussian noise, as shown in Fig. 3.7.

For Poisson noise, λ is set to 0.06. The mean value and variance of the Gaussian

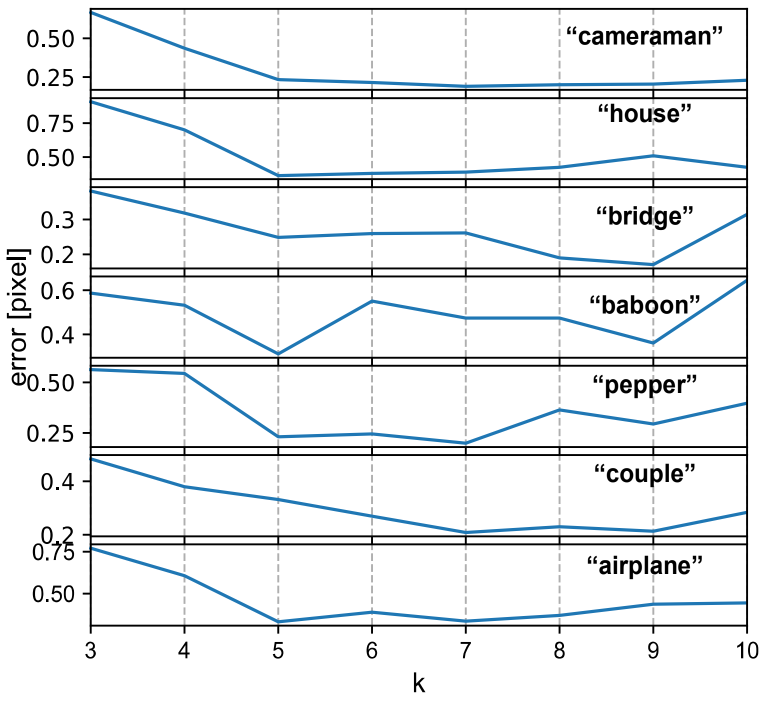


Figure 3.6: Parametric experimental results.

noise were set to 0 and 0.12, respectively. Furthermore, the image size was set to 64×64. The estimation results of the POC function was compared with those of the proposed MF-POC algorithm under the noise condition. The experimental result is shown in Fig. 3.8, wherein the 12 tested images re displayed from top to bottom, and each image is tested with the subpixel displacement from 0.0 to 0.9.

The POC function shows large estimation error peaks above 0.1 pixel around the displacements of 0.5 pixel and 0.7 pixel. However, in all tested images with different image content and subpixel displacement, the MFPOC algorithm maintained a smoother curve around 0, thereby effectively eliminating the error peaks appearing in the traditional POC. This experiment verified that even in real-world cases with some image noise, the proposed MFPOC algorithm can provide stable and high-accuracy measurement results.

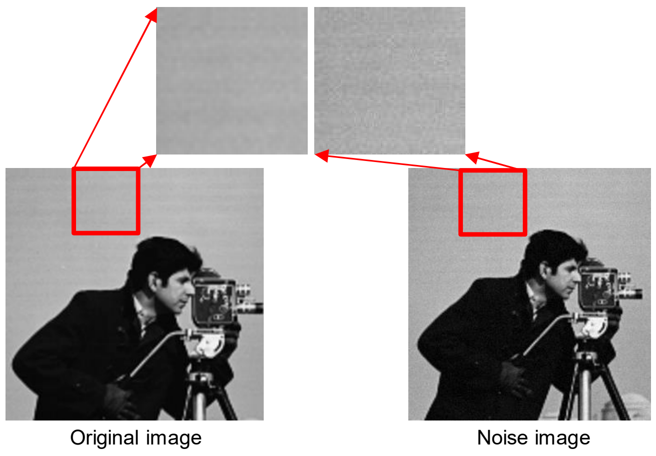


Figure 3.7: Noise image for simulating real cases.

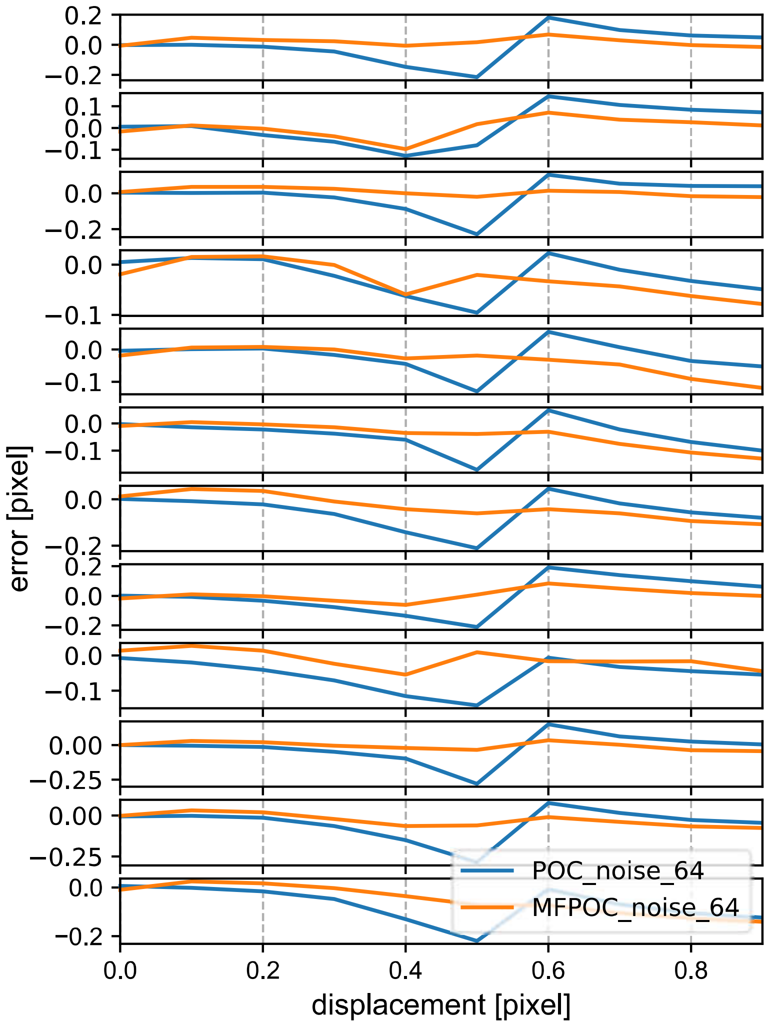


Figure 3.8: Experiment results for the noise images.

Chapter 4

Real-time Vibration Visualization Sensor Using GPU-based High-speed Vision

4.1 Introduction

Vibration monitoring is widely used in mechanical structure analysis [1–4] for operating machines with reciprocating and rotating components (e.g., engines, bearings, gear boxes, and motors) and structural health monitoring of civil infrastructures [5–7] (e.g., bridge condition analysis and building structure monitoring). Real-time vibration monitoring is critical for the continuous long-term monitoring of machines throughout their service life. Such monitored vibration data during operation can help operators identify early on damaged regions in structures, thereby reducing economic losses in machinery maintenance. Real-time and long-term monitoring of dilapidated buildings and bridges can help reduce accidents and unnecessary construction costs by extending their lifetime. In most cases, vibration monitoring is conducted by measuring the displacement, velocity, and acceleration using various types of contact sensors that need to be installed on the target structures to be observed [13, 15, 17]. These contact sensors can maintain the same motion with vibrating targets and provide accurate and robust vibration signals at their installation positions. These methods have two main limitations in vibration monitoring: (1) single-point measurement—multiple sensors need to be installed for a complex-shaped structure; however, dozens or hundreds of sensors require large installation times and increase system costs; and (2) uninstallable locations—there are many uninstallable locations for sensors in the cases of high-temperature, high-voltage, or small-scale structures operating at high speed.

Vision-based methods can address the aforementioned problems effectively. Cameras can remotely receive light signals reflected from the surfaces of vibrating structures as digital images, and the displacements at many measurement points can be calculated using computer vision algorithms [25] [28]. These methods can realize non-contact and full-field measurements without interfering with the targets to be observed. However, the limited image collection speed and high computational cost associated with image processing limit the sample frequency of the vibration signals in video-based monitoring. Conventional cameras working at 30 fps can only measure vibrations at frequencies below 15 Hz. In recent years, high-frame-rate (HFR) vision systems [35–37], which can capture and process images at hundreds or thousands of hertz, have been developed for the real-time tracking and recognition of fast-moving objects. Two main challenges still limit the realization of real-time full-field vibration monitoring: (1) Displacement measurements at thousands of points: faster computer vision algorithms are required for real-time full-field vibration monitoring as well as high-frame-rate capturing; (2) Real-time, human-readable monitoring: operators cannot directly observe vibration phenomena. High-speed phenomena are replayed offline in slow motion because they are too fast for the human eye to resolve.

In this study, we developed a real-time vibration visualization system that can estimate and display vibration distributions at all frequencies on a computer as real-time, human-readable data.

4.2 Concept of vibration visualization sensor

Most high-speed vision systems are designed for real-time sensing of high-speed scenes, and they function as software vibration sensors to output scalar image features in real time. However, captured HFR images can only be displayed in slow motion to operators on an offline computer display; thus there is a demand for real-time vibration visualization of HFR images as intuitive full-field images for long-term vibration monitoring.

To solve this problem, we introduce a two-step framework for real-time vibration visualization that involves HFR video processing to convert human-visible images at dozens of frames per second from invisible HFR images in parallel with full-field vibration displacement measurements. When input images of $M \times N$ pixels (frame number k) are obtained at time $k\tau$, denoted by $I(x, y, k\tau)$, the process flow is described below. τ and $f_0 = 1/\tau$ are the frame cycle time and frame rate, respectively.

(1) Full-field vibration displacement estimation

Input images of $M \times N$ pixels are converted to displacement images $\mathbf{A}(i, j, k\tau)$ of $M' \times N'$ pixels:

$$\mathbf{A}(i,j,k\tau) = \mathrm{Displacement}(I(x,y,k\tau))\ (k = 1,2,\cdots), \tag{4.1}$$

where the $M'N'$ displacement sensors are virtually located on the input images, and they are operated at the same sampling time as the camera cycle time τ. $\mathbf{A}(i, j, k\tau)$ $= (A_x(i, j, k\tau), A_y(i, j, k\tau))$ is composed of x- and y-displacement images with $M' \times N'$ pixels. Considering the computing resources for real-time execution, their resolutions are downconverted from $M \times N$ pixels.

(2) Conversion to temporal frequency response images

By performing STFTs of the displacement signals with K frames at all pixels in the displacement images $A_d(i, j, t + k'\tau)$ $(k' = 0,\cdots, K-1)$, the temporal frequency response

(TFR) images are computed as follows:

$$\begin{aligned}\mathbf{F}(i,j,t) &= (F_0(i,j,t),\cdots,F_{K-1}(i,j,t)) \\ &= \mathrm{STFT}(A_d(i,j,t),\cdots,A_d(i,j,t+(K-1)\tau)) \end{aligned} \tag{4.2}$$

where K determines the frequency resolution in STFT as $\Delta f = 1/(K\tau)$. $F_{k'}(x,y,t)$ indicates the frequency-component image at a frequency of $f_{k'} = k'\Delta f$ $(k' = 0,\cdots,K-1)$. The displacement image to be processed $A_d(i,j,t)$ is selected from the component images of $\mathbf{A}(i,j,t)$. The STFTs are performed at time $t = lT$ $(l = 1,2,\cdots)$, and T indicates the interval of the STFT computation. This corresponds to the frame-rate conversion for vibration visualization on a computer display. Owing to the symmetric frequency distribution in the FFT results, $K/2$ TFR images corresponding to $K/2$ frequency components are concatenated to generate a single image that presents all frequency information.

These TFR images, which are computed from the displacement images estimated at the camera frame rate, can indicate the vibration status as single-frame-based features. In this study, we computed the TFR images at all $K/2$ frequency bands, and simultaneously visualized them so that they could be observed by the human eye at an interval T of tens of milliseconds, which is relatively longer than the camera cycle time τ.

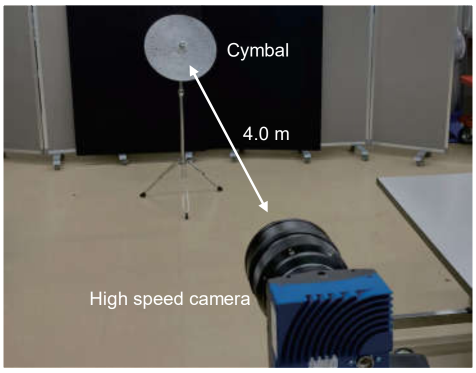
4.3 Proposed algorithm of frequency analysis

Based on high-speed GPU-based Batch POC, we can acquire full-field displacement images of $M' \times N'$ pixels at K consecutive frames, $A_d(i,j,t+k\tau)$ $(k=0,\cdots,K-1)$. For vibration analysis, these displacement images were simultaneously processed using STFTs, and their TFR images $\mathbf{F}(i,j,t) = (F_0(i,j,t),\cdots,F_{K\text{-}1}(i,j,t))$ were obtained with a frequency resolution of $\Delta f = 1/(K\tau)$ as the vibration visualization results. These operations were executed at intervals of T, which is relatively longer than the camera cycle time, τ. The displacement images computed in the DIC-based displacement measurement were transferred from the GPU to the CPU memory at intervals of τ, and the calculated displacement results at consecutive K frames were transferred from the CPU memory to the GPU for STFT computation. Finally, the TFR images processed on the GPU were transferred back to the CPU for vibration visualization every time the STFT computation starts at intervals of T.

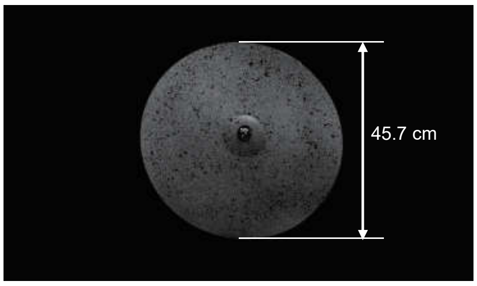
4.4 Vibration visualization system

For real-time vibration visualization, we used a GPU-based high-speed vision platform that can capture and process HFR images and display temporal frequency response images for full-field vibration displacements on a computer. It consisted of a high-speed CMOS camera head (EoSens 2.0CXP2, Mikrotron, Unterschleissheim, Germany) with a CoaXPress CXP-12 frame grabber (Coaxlink Quad CXP-12, Euresys, Seraing, Belgium) for HFR video capturing and a personal computer (PC) for HFR video processing accelerated by a GPU board.

The camera head had a 1920×1080-pixel CMOS image sensor(19.2×10.8 mm^2), with a pixel size of 10×10 μm^2. It could capture 8-bit gray images of 1920×1080 pixels at 2220 fps and transfer them to a PC using a CXP-12 frame grabber. We used a PC with the following specifications: ASUSTek WS C422 PRO/SE main board, Intel Core CPU i9-11700K @ 3.60 GHz, 10 cores, 128-GB memory, and Windows 10 Professional 64-bit OS (Microsoft, Redmond, WA, US). To accelerate HFR video processing, a GPU board (GeForce RTX 3090, NVIDIA, Santa Clara, CA, US) was installed on the PC.



(a) Experimental environment



(b) Observed cymbal

Figure 4.1: Experimental setup for a cymbal with free vibration.

4.5 Real-time vibration visualization experiments

4.5.1 Free vibration of a cymbal

To verify the effectiveness of our vibration visualization system, we show the experimental results of a metal cymbal with a diameter of 45.7 cm freely vibrated after knocking it with a wooden stick. Fig. 4.1 shows (a) a photo of the experimental setup and (b) a photo of the cymbal to be observed. The cymbal was vertically installed on a metal stand at a distance of 4.0 m from the camera head with an 85 mm lens. The cymbal was painted in a black and white speckled pattern. A total of 1920×1080 images ($M = 1920$, $N = 1080$) were captured and processed at 1000 fps ($\tau = 1$ ms). The block and step sizes in the DIC computation were set to $b_x = b_y = 64$ and $s_x = s_y = 64$, respectively. Corresponding to the 1920×1080 image, full-field displacement vectors of 30×16 blocks ($M' = 30$, $N' = 16$) were computed using DIC in real time at 1000 fps. In the experiment,

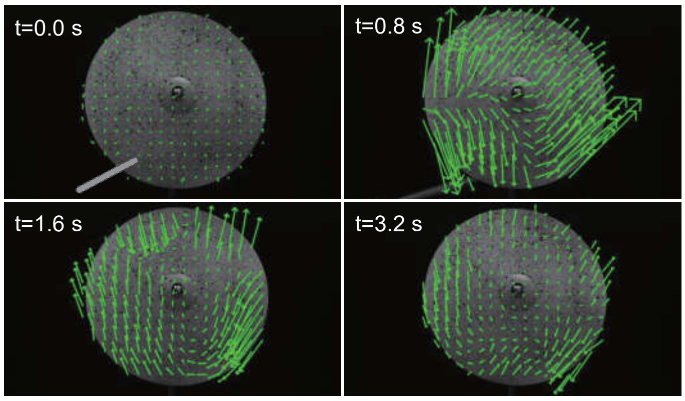
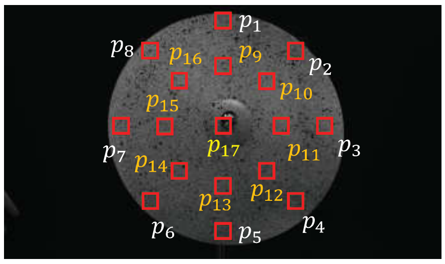


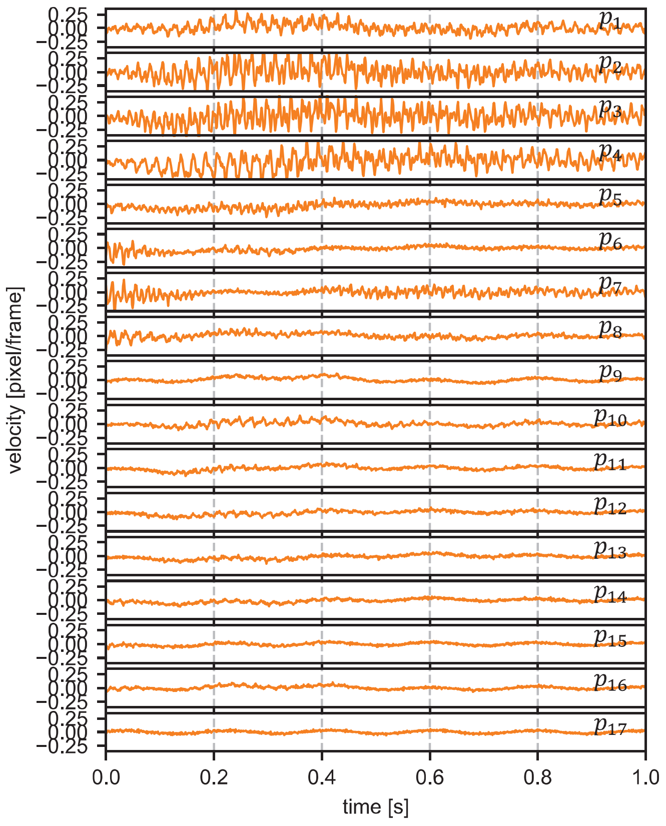
Figure 4.2: Estimated velocities in the cymbal experiment.

the DIC computation between $I_{ij}(x, y, t)$ and $I_{ij}(x, y, t - \tau)$, which were the sub-images at the current and previous frames ($t_R = t - \tau$), respectively, was executed to obtain full-field velocity vectors. A mask image $Q(x, y)$ was set to cover only the cymbal region in the images. The vertical velocity components were processed using STFTs with $K = 256$ at intervals of $T = 32$ ms; 128 TFR images in the range of 0–500 Hz with a frequency resolution of 3.9 Hz were displayed on a computer for vibration visualization in real time at 31.25 fps.

Fig. 4.2 shows the estimated velocity vectors at intervals of 0.8 s for $t = 0.0$–3.2 s. The cymbal was knocked by a wooden stick at $t = 0.6$ s, and the velocity vectors were magnified so that a 1-pixel-length corresponded to 2.0×10^{-3} pixel/s. Fig. 4.3 shows (a) the 17 measurement points on the cymbal, and (b) the vertical velocity components for $t = 1.0$–2.0 s. Fig. 4.4 shows their frequency amplitudes in the range of 0–500 Hz. As shown in Fig. 4.3(b), the vertical velocity components at $p_1, p_2, ..., p_8$ around the edge of the cymbal, vibrated more strongly than those at $p_9, p_{10}, ..., p_{16}$ located on the inner circle, and those at p_{17} around the center of the cymbal vibrated weakly with the smallest amplitude. Corresponding to the natural vibration of the cymbal, nonspatially uniform velocity vectors were time-varying, as shown in Fig. 4.3(b), whereas the peak frequencies in the frequency amplitudes were similar at the measurement points shown in Fig. 4.4.



(a) Measurement points



(b) Vertical velocity component .

Figure 4.3: Velocities at 17 measurement points on the cymbal.

The data shown in Fig. 4.5 were obtained for the vibration visualization data to be monitored on a computer display in real time at 31.25 fps. Fig. 4.5 shows (a) an image map of the 128 TFR images in the range of 0–500 Hz with their average frequency amplitude at t = 1.1 s, (b) TFR images at the peak frequencies of 42.9, 89.8, 148.4, and 210.9 Hz at t = 1.1 s, (c) the vertical velocity components at p_1 for t = 0.0–30.0 s, and

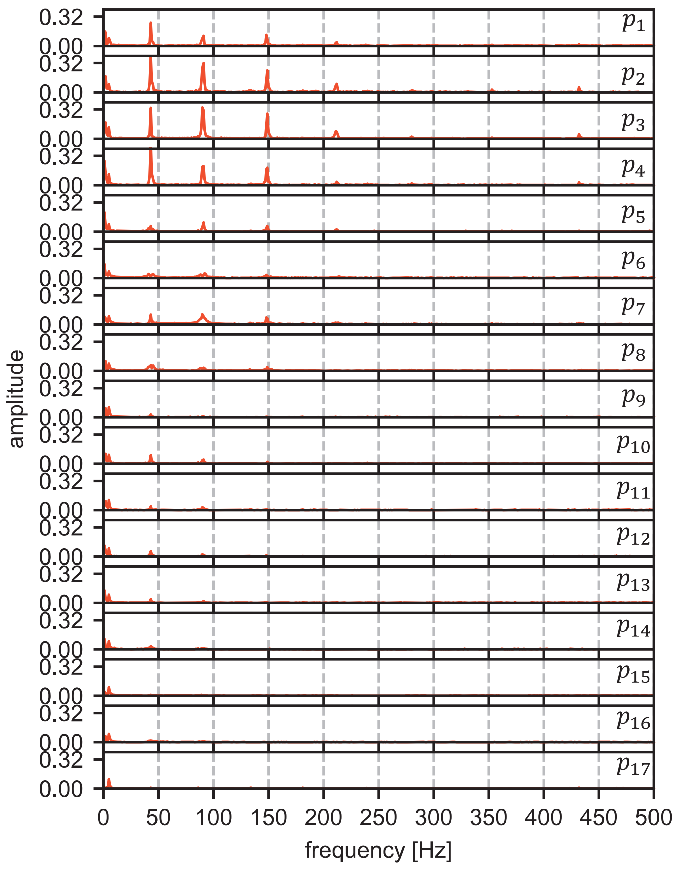
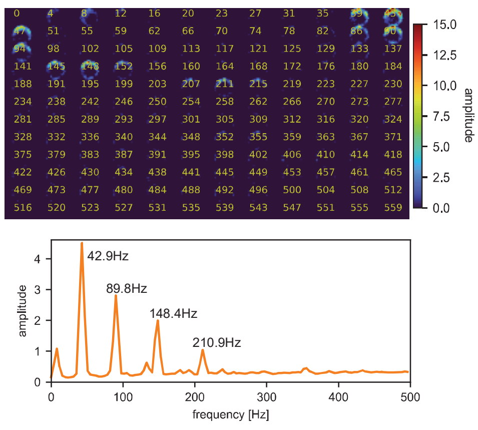
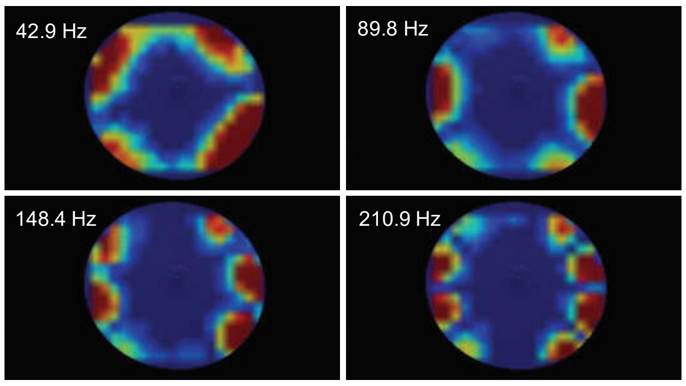


Figure 4.4: Frequency amplitudes at 17 measurement points on the cymbal.

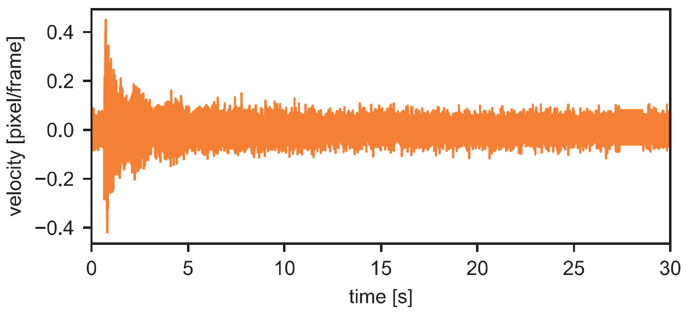
(d) the spectogram of the average frequency amplitude for t = 0.0–30.0 s. The colormaps in (b) and (d) indicate the frequency amplitudes. These peak frequencies correspond to the natural frequencies of the cymbal. The amplitudes of the TFR images correspond to various mode shapes. As shown in Fig. 4.5(d), frequency amplitudes at these peak frequencies were observed after knocking the cymbal at t = 0.6 s, whereas the frequency amplitude remained constant for a longer time as the peak frequency decreased. This is because the damping ratios of the vibration components in the low-frequency range were larger than those in the high-frequency range.



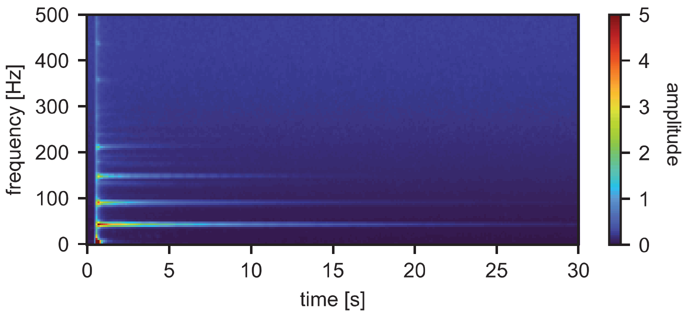
(a) TFR images and their averaged frequency amplitudes



(b) TFR images at peak frequencies

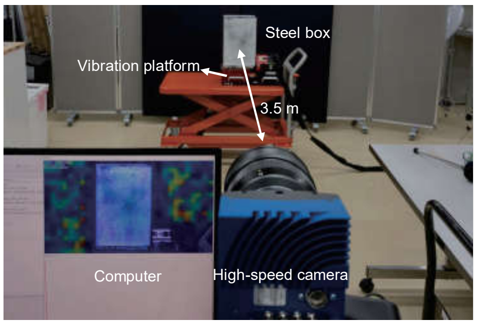


(c) Vertical vibration component at p_1 for t = 0.0–30.0 s

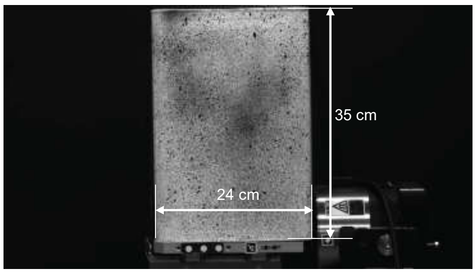


(d) Average vibration spectrogram for t = 0.0–30.0 s

Figure 4.5: Vibration visualization results in the cymbal experiment.



(a) Experimental environment



(b) Steel box to be observed

Figure 4.6: Experimental setup for a steel box with forced vibration.

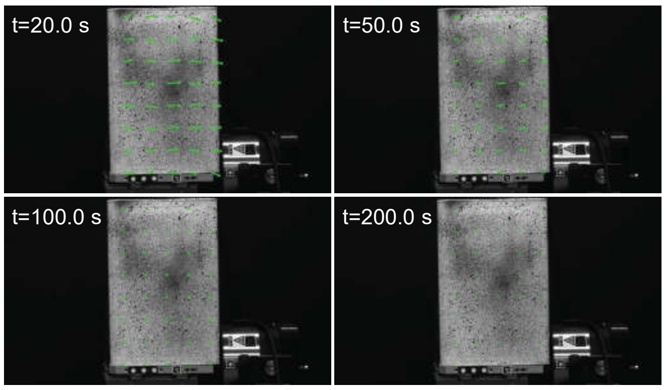


Figure 4.7: Estimated velocities in a steel-box experiment.

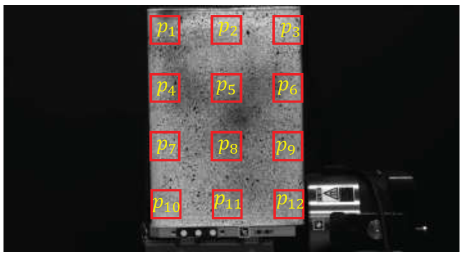
4.5.2 Forced vibration of a steel box

Next, we show the experimental results of a steel box forcibly vibrated on a vibration-testing machine. Fig. 4.6 shows (a) the photo of experimental setting and (b) the photo of steel box to be observed. A vibration-testing machine (D-Master APD-200FCD, Asahi Seisakusyo, Hino, Japan) excited the steel box in the horizontal direction by manually adjusting its sweeping frequency using a sine wave in the range of 5-200Hz for 272 s.

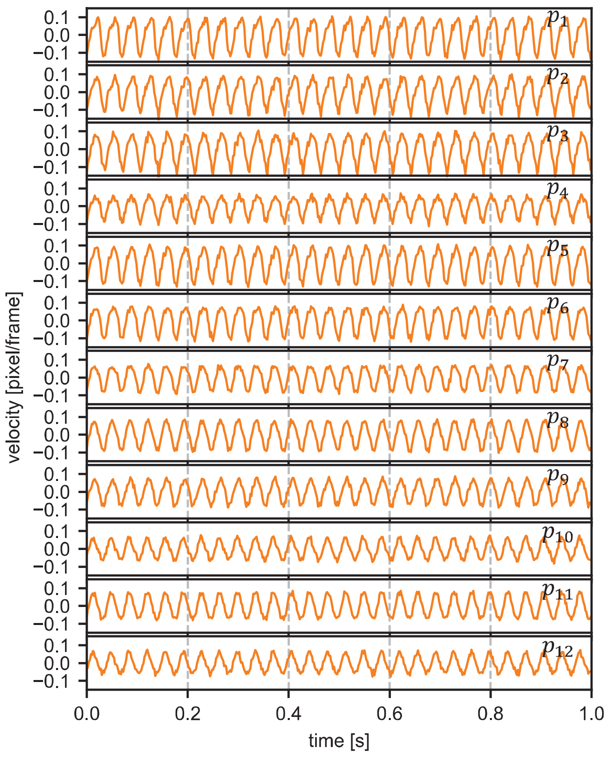
A steel box (35×24×24 cm) was painted in a black and white speckled pattern and fixed on the vibration testing machine using a belt. A camera head with an 85 mm lens was installed 3.5 m in front of the steel box so that its 35×24 cm^2 front surface could be observed in the images. With this installation, a 1920×1080 image and a pixel corresponded to 19.2×10.8 mm^2 and 10×10 μm^2, respectively. In the experiment, 1920×1080 images were captured and processed at 500 fps (τ = 2 ms), and the block and step sizes in the DIC computation were set to $b_x = b_y = 128$ and $s_x = s_y = 128$, respectively. With the DIC computation between the sub-images at the current and previous frames, full-field velocity vectors of 15×8 blocks ($M' = 15$, $N' = 8$) were computed in real time at 500 fps. A mask image $Q(x, y)$ was set to cover only the region of the steel box in the image. The horizontal velocity components were processed using STFTs with $K = 512$ at intervals of $T = 50$ ms. A total of 256 TFR images in the range of 0–250 Hz with a frequency resolution of 0.98 Hz were displayed in real time at 20 fps.

Fig. 4.7 shows the estimated velocity vectors at t = 20.0, 50.0, 100.0, and 200.0 s. The velocity vectors were magnified so that a 1-pixel-length corresponded to 2.0×10^{-3} pixel/s. Fig. 4.8 shows the (a) 12 measured points on the steel box and (b) horizontal velocity components for t = 30.0–31.0 s, when the metal box was vibrating at one of its resonant frequency of 28.3 Hz. Fig. 4.9 shows their frequency amplitudes in the range of 0–250 Hz. This shows that the metal box did not vibrate uniformly under the resonant frequency. Although all points were vibrating at the same peak frequency of 28.3 Hz, the amplitudes of p_1, p_2, p_3 distributed on the top region were larger than those of p_{10}, p_{11}, p_{12} distributed on the bottom region.

Fig. 4.10 shows the real-time visualized vibration data at three resonant frequencies of 28.3 Hz (t = 32.3 s), 45.9 Hz (t = 54.5 s), and 137.7 Hz (t = 177.1 s). The figure shows the (a) image maps of the 256 TFR images in the range of 0–250 Hz at the three moments and their average frequency amplitudes, (b) horizontal components at measurement point p_2 for t = 0.0–272.0 s, and (c) spectrogram of the average frequency amplitude at t = 0.0–



(a) Measurement points



(b) Horizontal velocity component

Figure 4.8: Estimated velocities at 12 measurement points on a steel box vibrating at 28.3 Hz.

272.0 s. When the sweep frequency increased over time, the amplitude of the horizontal component varied and the peak frequency increased, as illustrated in Fig. 4.10 (a) and (b). At the resonant frequencies when the steel box vibrated with a loud sound, the amplitudes of the TFR images at these frequencies corresponded to the various mode shapes shown in Fig. 4.10(a). The amplitudes were strong around its upper, middle-lower, and upper-edge sections in the 28.3 Hz-, 45.9 Hz-, and 137.7 Hz-TFR images, respectively.

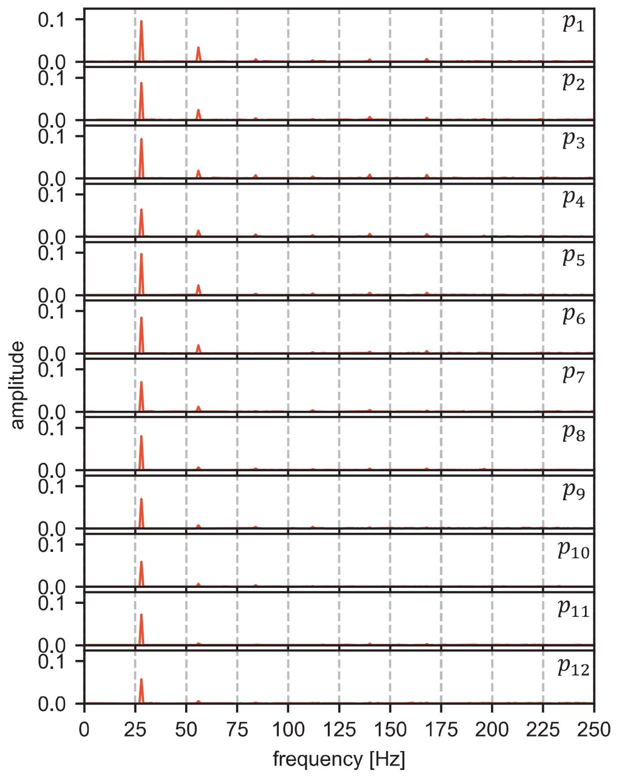
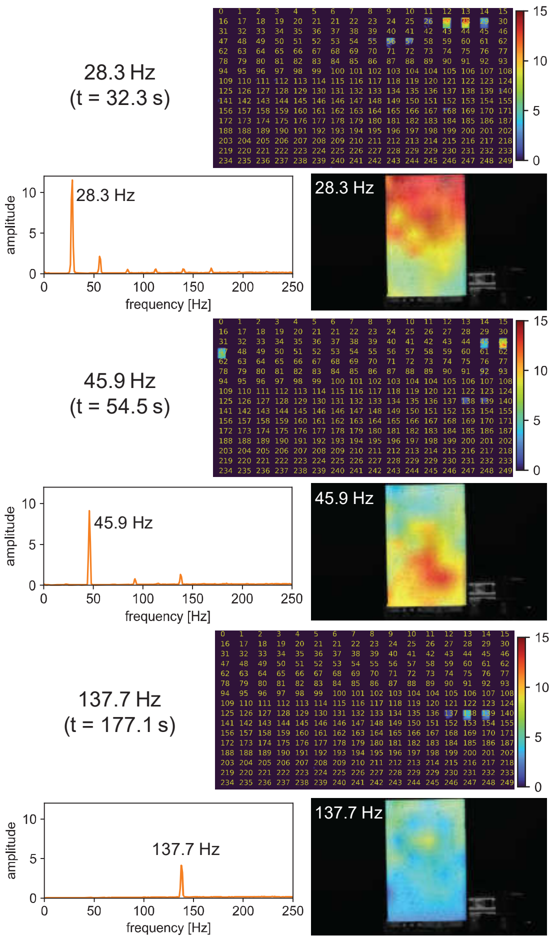


Figure 4.9: Frequency responses at 12 measurement points on a steel box vibrating at 28.3 Hz.

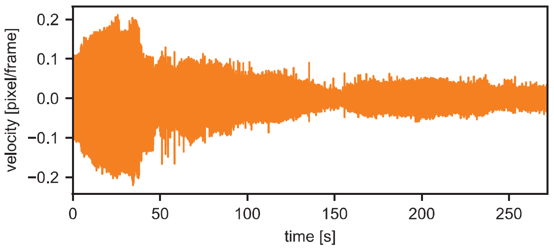
We confirmed that all data in Fig 4.10, as shown in Fig. 4.5, were monitored in real time on a computer display, and the resonant frequencies of the steel box and its mode-shape-like velocity amplitudes were intuitively visualized as vibration-feature data by using an image map of all frequency amplitudes. This can provide timely and effective information for structural analysis and mechanical defect detection to operators.

4.6 Concluding remarks

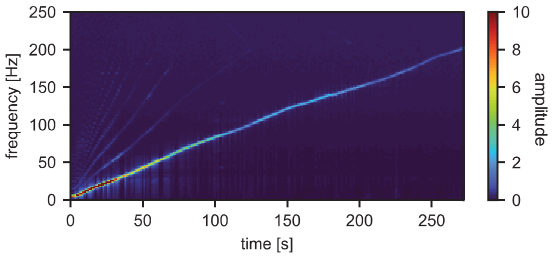
In this chapter, we realized real-time HFR-video-based vibration visualization in which operators can intuitively see the frequency responses of full-field vibration displacements at all frequencies as visible data at tens of frames per second on a computer display. Using parallel-accelerating DIC computations with STFTs on a GPU-based high-



(a) TFR images and their average frequency amplitudes



(b) Horizontal velocity component at p_2 for t = 0.0–272.0 s



(c) Average vibration spectrogram for t = 0.0–272.0 s

Figure 4.10: Vibration visualization in steel-box experiment.

speed vision platform, full-field vibration displacements of 1920×1080 images captured at 1000 fps were simultaneously visualized as TFR images in the range of 0–500 Hz. To verify its effectiveness, real-time monitoring experiments were conducted using a (1) free vibrating cymbal and (2) forced vibrating steel box. These experimental results demonstrate that our vibration visualization system can estimate full-field vibration responses at different levels of frequencies, which are extremely fast for human eyes to see. And operators can directly observe them as visible image data by displaying them at 31.25 fps on a computer.

This study focused on the real-time visualization of full-field vibration and utilized local information for DIC calculation, the accuracy depends heavily on the texture information of measured targets. For future work, we plan to improve the proposed system to generate more accurate velocity fields by merging the results obtained from multiple DIC calculations. Moreover, we aim to assess more practical applications of the system, such as bridge vibration monitoring and aided analysis for structural designs.

Chapter 5

Software Rotation Sensor Based on High-Speed Video Analysis

5.1 Introduction

In many factories, faults in rotating parts such as motors or gears can cause serious safety problems. Traditional measures such as post-event maintenance or regular inspections cannot provide timely abnormal signals for damage prediction.

In recent years, IoT sensors [79] have been used to capture continuous working signals of rotating machines for real-time inspection. However, their installation is limited by complicated machine structures, especially those with multiple rotating parts inside. High-speed vision systems operating at hundreds or thousands of frames per second can be utilized to capture and analyze high-speed motions not observable directly with naked eyes [80]. Several high-speed vision systems have been proposed to investigate periodic vibration faults [81–83], even though most of them are offline playback of less lengthy high-speed videos captured in slow motion and not utilized as real-time sensors. With the development of computational technology, several studies [84–86] have been conducted on high-speed vision that can achieve real-time capturing and processing of high-frame-rate (HFR) images, such as optical flow [87] and object tracking [88]. With vibrations at audio-level frequencies captured in real time, studies such as the analysis of the scraping behavior of experimental model mice [89] and drone rotating propeller detection [90] have been proposed, showing the effectiveness of using high-speed vision systems as real-time software sensors. These studies show that high-speed vision-based software sensors can also be applied in monitoring machine parts rotating at dozens of revolutions per second unseen by the naked eye.

Therefore, in this study, we propose a real-time software rotation sensor that can estimate the angles of rotational targets by processing high-speed video images containing multiple rotational objects in the same view. A parallel-implementing digital image correlation algorithm was developed to estimate the similarities between the input image and multiple reference images, corresponding to different angles at 500 fps. To verify its effectiveness, we conducted several experiments for multiple partially occluded rotating gears, showing that multiple rotation angles could be simultaneously estimated with high accuracy regardless of the posture and appearance of the target. Real-time angle estimation was also performed on a gear that rotated at 40 revolutions per second (2400 rpm).

5.2 Concept of rotation sensor

The proposed software rotation sensor is shown in Fig. 5.1. HFR-video is captured by a remotely installed high-speed camera. Rotation angles of multiple objects were estimated in real time by simultaneously calculating the similarities between current frame and numerous reference images at different angles. The software sensor has several advantages.

(1) Non-contact measurement without sensor mounting: Our software rotation sensor can be mounted without any interference on high-speed rotating objects that are difficult to attach. Its installation cost is relatively low and enables angular measurements at any position.

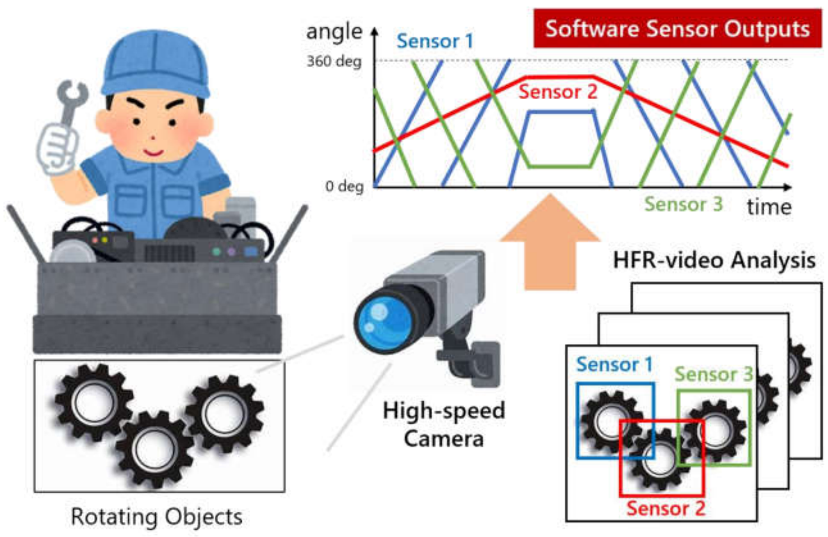


Figure 5.1: Concept of Software Rotation Sensor.

(2) Simultaneous measurement for multiple rotational objects: Every rotational object in the camera view is defined by a ROI region. By applying the software rotation sensor algorithm to each ROI image, our software rotation sensor can simultaneously estimate the rotation angles of multiple rotational targets.

(3) Robust measurement against posture and appearance changes: The relative position between the camera and the measured object does not change, ensuring that the visible part of the object maintains the same posture as in the image view. Our algorithm for software sensors is powerful against noise such as appearance changes and lighting fluctuations.

5.3 Proposed algorithm of parallelized similarity estimation

L reference images with M×N pixels were acquired in advance. Reference image $R_l(x, y)(l = 0, \dots, L-1)$ refers to rotation angle of $2l/L$. Rotational object is defined by a $M' \times N'$ ROI image $R'_l(x', y') = R_l(x + x_0, y + y_0)$. The following steps were executed frame-by-frame:

(1) High-speed input image and ROI image obtention

Input image at moment t is recorded as $I(x, y, t)$, and ROI image corresponding to the inspected object is cropped as $I'(x', y', t) = I(x + x_0, y + y_0, t)$. The frame interval is Σt (frame rate $F = 1/\Sigma t$).

(2) Correlation between input ROI and reference ROI

The correlation value $C_l(t)$ between the reference ROI image $R'_l(x, y)$ and input ROI image $I'(x', y', t)$ is calculated using the following equation:

$$C_l(t) = \mathrm{DIC}(I'(x', y', t), R'_l(x', y')). \tag{5.1}$$

Here, digital image correlation is achieved using the phase-only correlation method [13]. At t=0.0 s, correlation values between every L reference ROI image ($l = 0, \ldots, L-1$) and input ROI image are calculated to obtain the initial angle index estimation. Then, based on the estimated angle index of the last frame $l_p(t - \triangle t)$, only L' reference images ($l \in N(l_p(t-\triangle t))$) whose angle indices are close to $l_p(t-\triangle t)$ will be sought in the following correlation calculation.

(3) Angle index detection based on correlation peak value

The matched angle index $l_p(t)$ is detected by determining the maximum correlation value, $C_l(t)$. When $t = 0$, all reference images are searched, or only reference images near the detected angle index of the last frame $l_p(t - \triangle t)$ will be.

$$l_p(t) = \begin{cases} \max_l C_l(t), t = 0 \\ \max_{l\in(l_p(t-\triangle t))} C_l(t), otherwise \end{cases} \tag{5.2}$$

(4) Rotation angle estimation based on parabolic fitting

The correlation values $C_l(t)$ of three angle indexes namely: the peak angle index

$l_p(t)$ and those before and after it $l_p(t)-1, l_p(t)+1$, are utilized to execute parabolic fitting for higher estimation accuracy of rotation angle $\phi(t)$than the angle interval of reference images.

$$\phi(t) = \frac{\pi(C_{l_p(t)-1}(t) - C_{l_p(t)+1}(t))}{L(C_{l_p(t)-1}(t) - 2C_{l_p(t)}(t) + C_{l_p(t)+1}(t))} + \theta(l_p(t)) \tag{5.3}$$

Here, $\theta(l_p(t))$ refers to the angle value that corresponds to angle index $l_p(t)$.

5.4 Rotation analysis system

To verify the effectiveness of our software rotation sensor, we configured a GPU-based vision system that could capture 1920×1080 images at 1000 fps or higher frame rates. The system comprises a high-speed camera head (EoSens 2.0CXP2, Mikrotron) with a Coaxlink Quad CXP-12 frame grabber for high-speed image transfer, a personal computer (Dell, USA) equipped with an Intel Core CPU i9-9900K @ 3.60GHz, 32GB-memory, a GPU board of NVIDIA Geforce RTX 2080Ti (NVIDIA, Santa Clara, US), and Windows 10 Professional 64-bit OS (Microsoft, Redmond, WA, US).

For the parallelized algorithm on the GPU, the reference image count L is set at 360 and that of $L^{'}$ at 8. Execution times of DIC for different sizes of ROI images are shown in Table 5.1. For 128×128 ROI calculation, the angle estimation for the first time was 8.39 ms and the next was 0.97 ms. Compared to CPU-only-based processing, with 68.26 ms (the first-time estimation) and 2.25 ms (the following estimation), the processing speed values were 8.1 and 2.3 times faster, respectively, indicating that our software rotation sensor can estimate rotation angles in 1 ms and process HFR-video at 1000 fps in real time.

Table 5.1: Execution time (unit: ms)

parameters	64×64	128×128	256×256	512×512
first frame	2.41	8.39	55.16	327.97
others	0.71	0.97	2.08	5.38

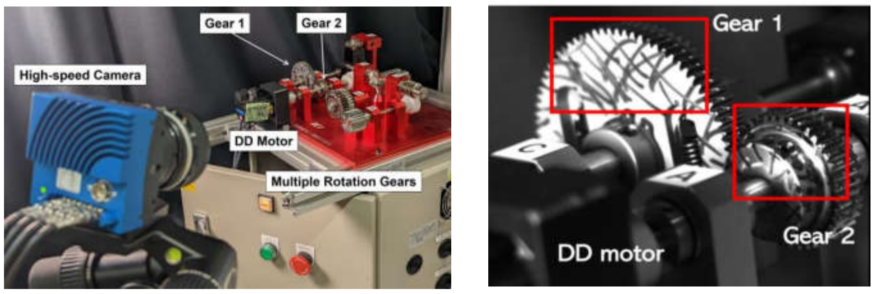


Figure 5.2: Left: Experimental Settings, right: Gears to be inspected

5.5 Experiments for rotation analysis

In this section, we show the utilization of multiple rotary gears as observation targets for experimental verification. Fig.5.2 shows the experimental settings including the camera and an example image of multiple gears. The KG gear education kit (Kyoiku Gear Industry) is driven by a DD motor (SGM7A-C2AFA21, Yaskawa Electric). Gear 1 (80 mm diameter, 80 teeth) and 2 (43 mm diameter, 40 teeth) with different reduction ratios were photographed diagonally at high speed with a resolution of 480×480 pixels. Gear 2 was 68 cm away from the camera with a physical resolution of 0.14 mm per pixel. Gear 2 is directly connected to the DD motor and rotates at the same rotation speed as the motor, whereas gear 1 rotates at half the rotation speed of gear 2. Each gear was patterned to improve the accuracy of the rotation angle measurements by reducing the mismatch in the DIC calculation. The count of the reference images was set as L= 360 in the following experiments.

5.5.1 Offline Analysis for Multiple Rotary Gears

To confirm the measurement accuracy of our proposed software rotation sensor, we recorded a 6-second HFR video of gears 1 and 2 with a resolution of 800×800 at

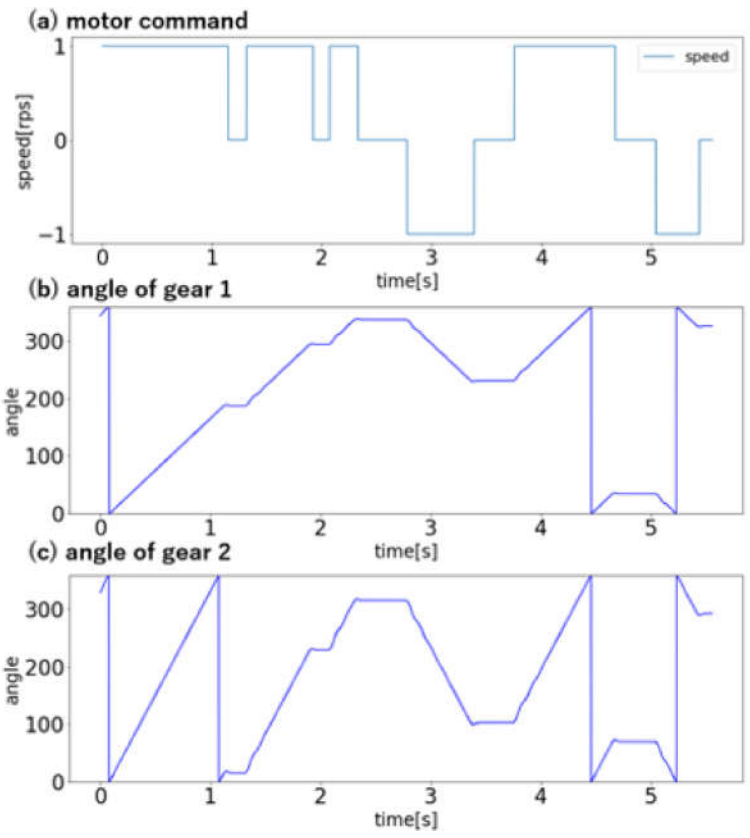
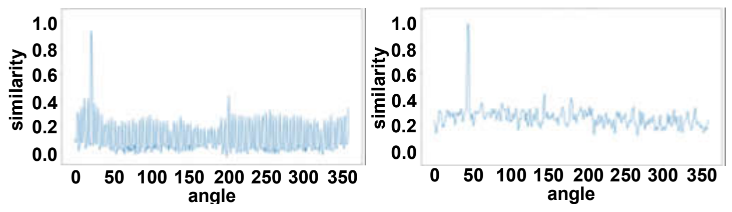


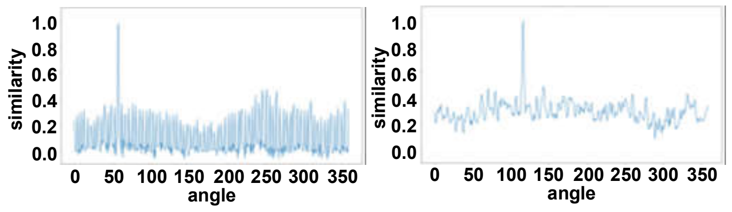
Figure 5.3: Motor command and estimated angles in multi-gear experiment.

1080 fps for offline analysis when the forward, reverse, and stop commands at 1 rps were intermittently sent to the DD motor.

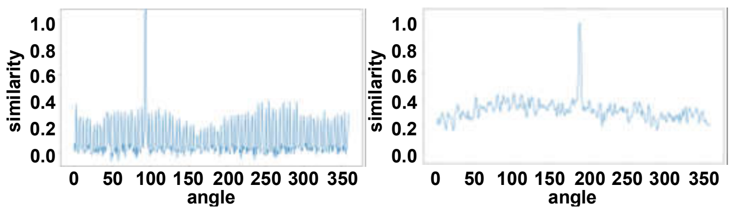
Fig. 5.3 shows the (a) rotation command values during t= 0.0 6.0 s, (b) rotation angle of gear 1, and (c) rotation angle of gear 2. The rotation angles of gears 1 and 2 increased to 180° and 360° respectively during the recorded t= 0.0 1.0 s. The rotation speed of gear 1 was 0.5 times that of gear 2, corresponding exactly to the reduction ratio. In Fig. 5.4, the left and right sides show the correlation similarity curves for gears 1 and 2, respectively. The correlation similarity curves of gears 1 and 2 both showed a remarkable peak, with their peak positions moving by 36° and 72°, respectively, every 0.2 s. These movements correspond to the rotation of gears 1 and 2 at 180 deg/s and 360 deg/s, respectively. The accuracy of the rotation angle estimated at the 0.1-deg level greatly



(a) t = 0.2 s



(b) t = 0.4 s



(c) t = 0.6 s

Figure 5.4: Similarities for 360 reference images.

exceeded the 1-deg step of the reference images after applying parabolic fitting. These results revealed that with a high-speed camera working as a software rotation sensor, the rotation angles of multiple gears could be simultaneously and accurately measured, even when the gears were partially occluded.

5.5.2 Real-time Experiments for High-speed Rotary Gears

For the real-time experiment, the scene of gear 2 rotating at 40 rps was monitored using a camera with a resolution of 480×480 at 500 fps, wot a 128×128 ROI region corresponding to gear 2 processed in real time. The correlation similarity values between the 65 reference images ($L' = 65$, 32 reference images before and after the last matched reference image) and the current frame were calculated parallelly. The maximum rotation speed ω_{max} that our software sensor could measure was determined by the frame rate F,

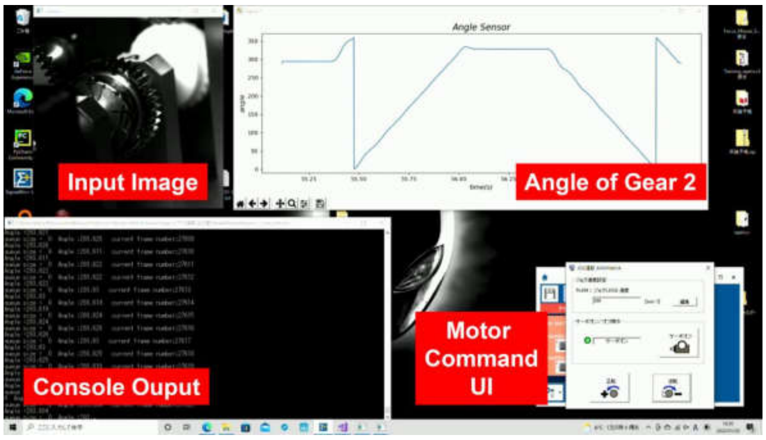


Figure 5.5: Screen capture in real-time experiment.

reference image count L, and searched reference image count $\omega_{max} = (L' - 1)F/2L$.

According to the above-mentioned parameters, our sensor can measure rotation speeds up to 44.4 rps (2664 rpm). Fig. 5.5 shows a screenshot of a personal computer screen where the software rotation sensor was operated in real time. The upper left is the input image of the gears (480×480), the upper right is the time-changing estimated rotation angle, and the lower right is the user interface for the DD motor command. When electrical commands were intermittently sent to the DD motor through screen operations, high-speed rotation at 40 rps was measured in real-time, confirming that our software rotation sensor could visualize rotation by monitoring the time-changing rotation angle in a manner similar to an oscilloscope.

Fig. 5.6 shows the real-time experimental results: (a) rotation command values for the DD motor and (b) the time-changing rotation angle of gear 2. During t= 46.0 47.0 s, the rotation angle increased from 0 deg to 360 deg 40 times, corresponding to the rotation speed of gear 2. From t= 47.7 s, the rotation angle decreased from 360° to 0° 40 times, owing to the reverse command. During the quick start of gear 2, sliding friction with interlocking gear 1 occurred, and abnormal vibrations of the rotation angle were observed for a short time. This experiment revealed that our developed software rotation sensor could measure high-speed rotations up to 2400 rpm with high accuracy in real

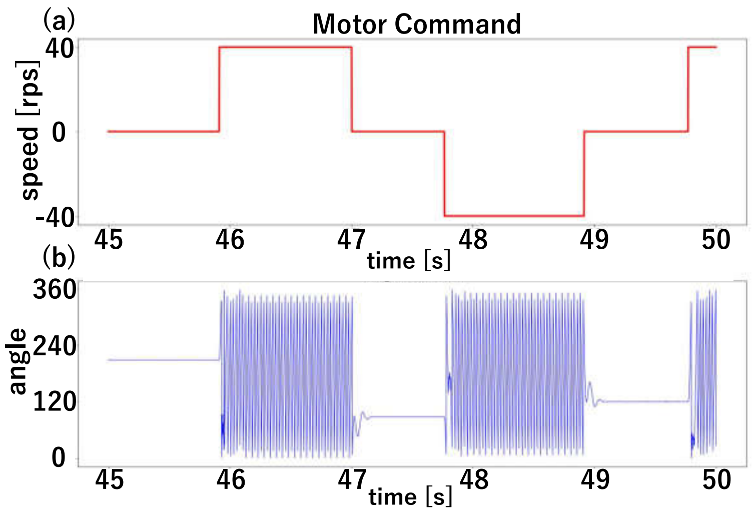


Figure 5.6: Motor command and estimated angle in real-time experiment.

time.

5.6 Concluding remarks

In this study, we propose a software rotation sensor that can estimate the rotation angles of multiple rotary objects in a high-speed video. The experiment on multiple gears showed that one high-speed camera could function as a multiple software rotation sensor, and rotation angles could be accurately estimated. Moreover, a real-time experiment confirmed the effectiveness of the software rotation sensor, with the rotation of a high-speed gear rotating at 40 rps successfully monitored in real time.

Chapter 6

HFR-video-based Fingertip Velocimeter for Multi-finger Tapping Detection

6.1 Introduction

A human finger is capable of rapidly moving with dexterity, and many traditional input interfaces, such as computer keyboards and mice, have been designed on the presumption of finger mobility [91]. With the rapid progress of human-computer interaction technologies, there is a strong demand for fingertip tracking as an unconstrained human interface that allows an operator to freely control computer systems and devices using their fingers. Recently, fingertip-tracking-based interfaces have been adopted in many fields, such as virtual reality [92–94], augmented reality [95–97], robot control [98–100], and IoT device control for smart households [101–103].

Existing fingertip interfaces mainly include two categories: sensor-mounted fingertip interfaces that require sensors on fingertips or targets to be touched and camera-based fingertip interfaces that capture fingertip motions with remote cameras.

The most popular sensors for sensor-mounted fingertip interfaces are inertial measurement unit (IMU) sensors [104–106] that can directly output finger acceleration information in three orthogonal directions. Magnetic sensors [107–109] can realize mid-air fingertip tracking based on the interaction between a magnet placed on the fingernail and magnetometer array under the device. Stretch sensors [110] [111] are usually mounted to obtain finger motion based on the bending angle. Sensor-mounted interfaces can provide accurate finger motion signals in real time regardless of vision occlusion and complicated motions. However, mounted sensors may cause stress on fingers and non-user-friendly interactions. Moreover, they are not always suitable for multiple finger measurements because of their complicated installment and management.

In camera-based approaches, finger motions are tracked using real-time video processing without mounting any auxiliary device on the user. Several studies [112–114] have been reported for hand-region segmentation with color- and contour-based features but they are not always stable when implemented with time-varying illumination and background interferences. RGB-D cameras, such as Kinect and Leap Motion Controller [115–119], can solve this problem by extracting hand regions with depth information but they cannot observe objects at long distances because of the power limitations regarding emitted beams [120]. Recently, many deep learning methods [121–124] have been proposed for image-based finger detection in natural scenes and achieved state-of-the-art performance, mainly focusing on detecting and tracking the positions of fingertips in standard videos at dozens of frames per second. These standard camera-based studies cannot provide precise fingertip velocities or accelerations to analyze finger motion that involves quick tapping at dozen times in a second [125].

To capture high-speed phenomena at the audio-frequency level in real-time, high-speed vision techniques such as vision chips have been widely used in the field of computer vision. Compared to conventional image processing speeds of dozens of frames per second, high-speed vision systems operating at hundreds or thousands of frames per second can simultaneously capture and detect human-invisible fast movements, and their effectiveness has been demonstrated in various applications such as robot control [126], vibration measurement [127] and quadcopter navigation [128]. If such high-speed vision can capture and detect finger motion with quick finger tapping, it would become a key device for next-generation human interfaces without mounting any auxiliary devices on users.

In this study, we propose a novel concept of a software-based fingertip velocimeter that can estimate when and where fingertips tap by executing high-frame-rate (HFR) video processing. This concept enables fingertip velocity estimation with sub-pixel precision in real time by hybridizing digital image correlation (DIC) [129] at hundreds of frames per second with CNN-based object detection operating at dozens of frames to determine the image regions of fingertips as software sensor positions. We implemented a software-based fingertip velocimeter algorithm on 720×540 resolution images operating at 500 fps; its effectiveness was demonstrated by performing several finger tapping experiments, such as a real-time virtual keyboard experiment with multi-fingertip tapping at a dozen times in a second.

6.2 Concept of fingertip velocimeter

In this study, we proposed a concept named software-based velocimeter that enables dynamic velocity analysis of moving objects in HFR images by hybridizing the high robustness of AI-based object detection and high accuracy of sub-pixel DIC measurements. Fig. 6.1 summarizes the advantages and disadvantages of AI detection, DIC estimation, and our concept. Fig. 6.2 presents how our proposed software-based velocimeter generates high-frequency velocity signals for dynamic analysis. Given an HFR video as input, AI detection is executed at a longer time interval to robustly update target locations in real time, and DIC is performed frame-by-frame to accurately estimate velocities with the same frequency as the input video. The recorded high-frequency velocity signals can be used for dynamic analysis in different applications.

Method / Characteristic	AI Detection	DIC Estimation	Ours
Target Selection	Automatically	Manually	Automatically
Robustness	High	Low	High
Accuracy	Several Pixels	Subpixel	Subpixel
Speed	Slow	Fast	Fast

Figure 6.1: Advantages and disadvantages of AI detection, DIC estimation, and our concept.

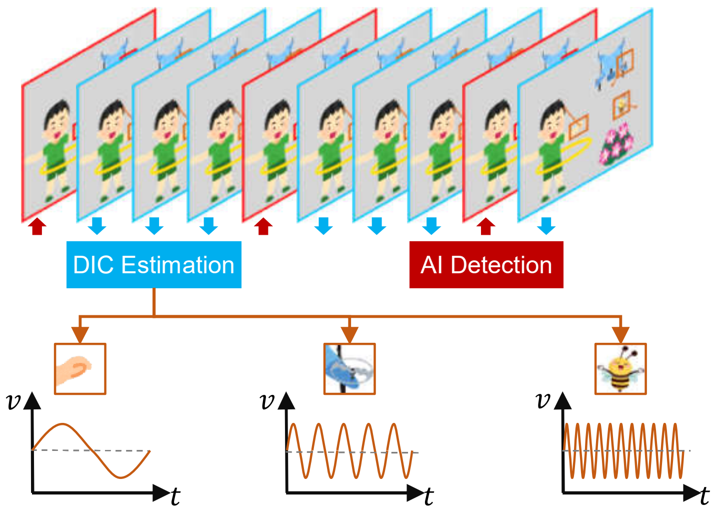


Figure 6.2: The proposed software-based velocimeter for dynamic analysis.

6.3 Proposed algorithm of finger tapping analysis

To verify the effectiveness of our concept, we developed a software-based fingertip velocimeter for computer interaction that can simultaneously detect tapping positions and timings of multiple fingers by estimating their velocities at high frequencies in real-time. Fig. 6.3 shows the framework of our fingertip velocimeter. Given an HFR image sequence, $(I(x, y, k\tau_h)$ denotes an image of $M \times N$ resolution captured at $t = k\tau_h$, where τ_h denotes the frame cycle time. The procedure for our fingertip velocimeter is divided into the following three stages.

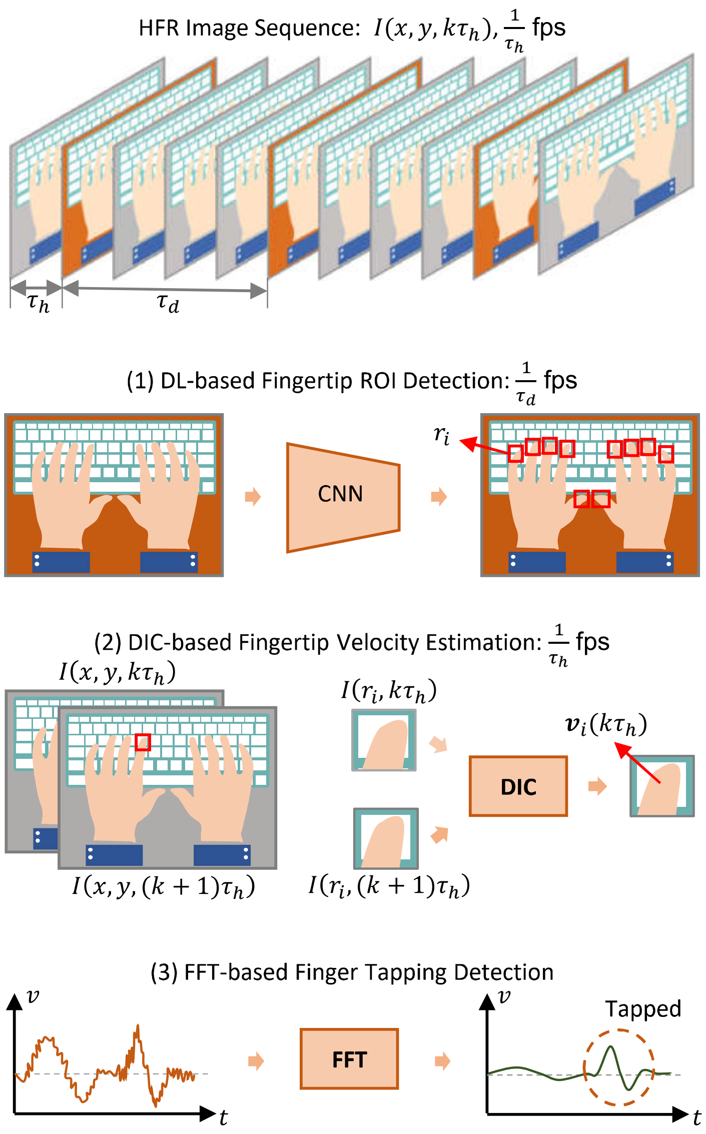

HFR Image Sequence: $I(x, y, k\tau_h), \frac{1}{\tau_h}$ fps
τ_h
τ_d
(1) DL-based Fingertip ROI Detection: $\frac{1}{\tau_d}$ fps
CNN
r_i
(2) DIC-based Fingertip Velocity Estimation: $\frac{1}{\tau_h}$ fps
$I(x, y, k\tau_h)$
$I(x, y, (k+1)\tau_h)$
$I(r_i, k\tau_h)$
$I(r_i, (k+1)\tau_h)$
DIC
$\boldsymbol{v}_i(k\tau_h)$
(3) FFT-based Finger Tapping Detection
v
t
FFT
Tapped

Figure 6.3: Framework of software-based fingertip velocimeter.

6.3.1 DL-based Fingertip ROI Detection

To localize the multiple fingertips in images, the MediaPipe network [130] is used for fingertip detection as a DL-based hand-landmark detection method. It can extract 21 landmarks from each hand and each finger is represented by four landmarks $\{\boldsymbol{p}_i^1, \boldsymbol{p}_i^2, \boldsymbol{p}_i^3, \boldsymbol{p}_i^4\}$ from its base to its end, as shown in Fig.6.4.

Corresponding to these fingertip landmarks, a fingertip ROI $\boldsymbol{r}_i$ is set for the fingertip

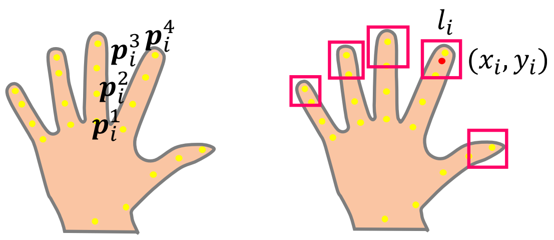


Figure 6.4: Locations of fingertip ROIs.

of the *i*-th finger ($i = 0, \cdots, 4$) as a square sub-image region; with side length l_i and center position $\boldsymbol{x}_i = (x_i, y_i)$ determined as follows:

$$l_i = k_1 \cdot |\boldsymbol{p}_i^4 - \boldsymbol{p}_i^3|, \tag{6.1}$$

$$\boldsymbol{x}_i = k_2 \cdot \boldsymbol{p}_i^4 + (1 - k_2) \cdot \boldsymbol{p}_i^3. \tag{6.2}$$

where k_1 is a coefficient for setting the side length sufficiently large so that the background scene is not largely involved in the fingertip ROI, and k_2 is a coefficient for setting the center position so that pixels occupied with a finger are largely involved in the fingertip ROI when its end is observed in the ROI. We can reduce inaccurate estimation of fingertip velocity by properly setting these parameters to include the fingertip area largely in the ROI.

6.3.2 DIC-based Fingertip Velocity Estimation

Given the ROI region $\boldsymbol{r}_i$ of fingertip i, its velocity $\boldsymbol{v}_i$ is obtained by executing the phase-only correlation (POC) algorithm [131] at a time interval of τ_h,

$$\boldsymbol{v}_i = \mathrm{POC}\left(I\left(\boldsymbol{r}_i, k\tau_h\right), I\left(\boldsymbol{r}_i, (k+1)\tau_h\right)\right) \tag{6.3}$$

$\boldsymbol{v}_i = (v_x^i, v_y^i)$ denotes the velocity vector of the i-th fingertip. $I(\boldsymbol{r}_i, k\tau_h)$ refers to the cropped fingertip ROI $\boldsymbol{r}_i$ from input frame $I(x, y, k\tau_h)$.

POC can output sub-pixel-accuracy estimated velocities by interpolating integer results calculated from the pixel-level similarity distribution in the frequency domain. Moreover, owing to the small sub-image size and FFT-based acceleration, POC can be executed at a millisecond-level speed to realize HFR velocity measurement.

6.3.3 FFT-based Finger Tapping Recognition

Finger tapping can be divided into non-contact tapping, corresponding to taps in the air, and contact tapping, corresponding to contacting a plane. Contact tapping is more in line with human operating habits because of the support and feedback provided by the physical plane. When a finger contacts a plane, its velocity suddenly stops, and high-frequency components are generated in the frequency domain. Thus, contact tapping can be distinguished from non-contact tapping by extracting the frequency components generated by the contact action.

The camera view is another key factor in vision-based fingertip motion measurement. Fig. 6.5 shows the hand images captured under different camera views. The side view provides a larger motion amplitude but blurs the fingertip positions on the plane. The top view can easily monitor clear finger trajectories at the expense of smaller tapping amplitudes. Moreover, finger tapping can still be detected because it is not a strict vertical reciprocating motion.

Our finger velocimeter was designed to detect effective contact-tapping actions under the top camera view. Fig. 6.6 shows the data measured under the top view: (a) the vertical velocity component of non-contact tapping, (b) the vertical velocity component of contact tapping, (c) their frequency distribution, and (d) bandpass filtered tapping signals.

The peak width in contact tapping is approximately half of that in non-contact tap-

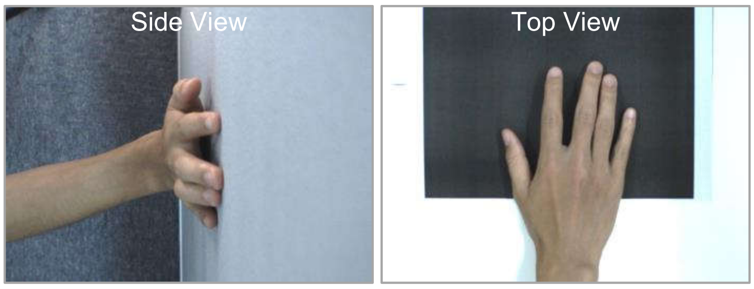


Figure 6.5: Hand images captured under different camera views.

ping because the contact action suddenly stops the fingertip. In the frequency domain, the non-contact tapping signal has stronger low-frequency components in the range of 0–10 Hz, which corresponds to the basic tapping circle. Except for the basic tapping frequencies, the contact tapping signal exhibits larger amplitudes in the frequency range of 20–50 Hz, which is generated by the contact action. From the filtered signals with only 20–50 Hz frequency components remaining, observe that the contact tapping signal retains obvious peaks, whereas the non-contact tapping signal retains only random noise.

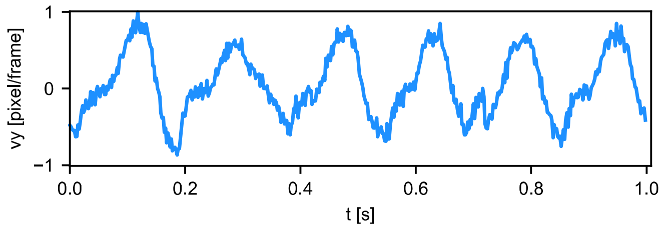
Based on the results of the tapping analysis, N-frame velocity signal $\tilde{V}_i$ is filtered by the following equations to make it more discriminative for contact-tapping detection.

$$F_i = \mathcal{F}_{1d}(V_i), \tag{6.4}$$

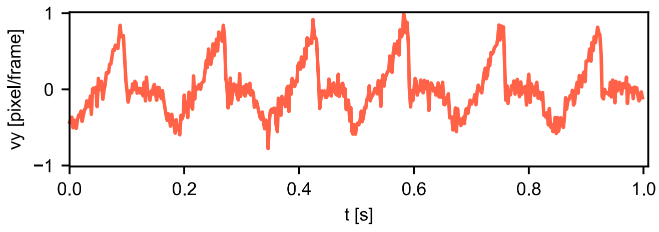
$$F_i^{'}(k) = \begin{cases} F_i(k),\ k_1 \leq k \leq k_n \\ 0,\ \text{otherwise} \end{cases}, \tag{6.5}$$

$$\tilde{V}_i = \mathcal{F}_{1d}^{-1}\left(F_i^{'}\right). \tag{6.6}$$

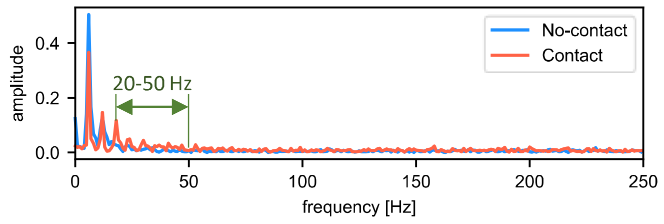
where $\mathcal{F}_{1d}$ and $\mathcal{F}_{1d}^{-1}$ refer to the 1-D FFT and inverse FFT operations, respectively. k is the index of the k_{th}-frequency component. k_1 and k_n represent the frequency components



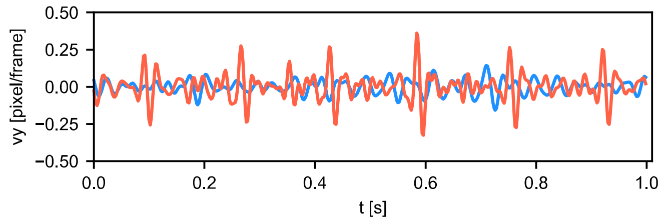
(a) Vertical velocity component of the non-contact tapping



(b) Vertical velocity component of the contact tapping



(c) Frequency distribution of the two kinds of tapping signals



(d) Bandpass filtered taping signals in the range of 20–50 Hz

Figure 6.6: Tapping analysis results from the top view.

at 20 Hz and 50 Hz, respectively. $\tilde{V}_i$ represents the filtered velocity sequence $\{v_1, \cdots, v_N\}$ used for tapping recognition.

Here, N refers to the signal length selected for tapping detection. A signal that is too short will miss the complete tapping shape and cause low-frequency resolution, which makes it difficult to select the expected frequency range. A signal that is too long may include two or more tapping actions, which can confuse peak detection. Thus, N should be properly set according to the tapping speed.

Based on the filtered velocity signal $\tilde{V}_i$, we designed the following two constraints to locate the tap position.

(a) Double-peak constraint: A complete finger-tap signal should contain a positive peak P_p and negative peak P_n. Each peak can be defined using five points.

$$P_p = p,\ if \left(v_p = \max\left(v_{p-5}, v_{p-1}, v_p, v_{p+1}, v_{p+5}\right)\right), \tag{6.7}$$

$$P_n = n,\ if \left(v_n = \min\left(v_{n-5}, v_{n-1}, v_n, v_{n+1}, v_{n+5}\right)\right). \tag{6.8}$$

Here, P_p and P_n denote the indices of the frames where positive and negative peaks occur, respectively.

(b) Peak shape constraint: The magnitudes of P_p and P_n should be larger than the peak threshold A_{TH} to ensure sufficient tapping amplitude. Their time interval should not be shorter than T to further filter out similarly shaped noise signals.

$$v_{P_p} > A_{TH}, \tag{6.9}$$

$$v_{P_n} < -A_{TH}, \tag{6.10}$$

$$P_n - P_p > T. \tag{6.11}$$

At the frame when the negative peak is detected, all conditions are calculated to determine whether the contact-tapping action occurs.

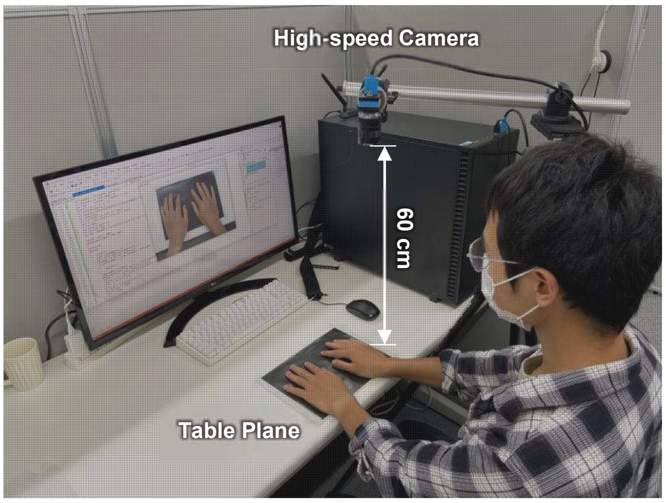


Figure 6.7: The system configuration of finger tapping detection.

6.4 Finger tapping system

To verify the performance of our fingertip velocimeter, we developed a multi-finger tapping detection system that can measure fingertip velocities at 500 fps and output the tapping signals of multiple fingers in real time. As Fig. 6.7 demonstrates, the system consists of a high speed CMOS camera (Image Source Camera, DMK37BUX287, made in Germany) equipped with a 1/2.9 inch image sensor (IMX287LQR, SONY, Japan) for high-frame-rate video capturing and computer for performing high-speed DIC computations. The high-speed camera was installed above the table plane and could capture RGB images of 720×540 pixels at 500 fps and transmit them to the computer via the USB 3.1 protocol. The computer runs a Windows 10 (64-bit) operating system with an Intel Core i7-8750H 2.2GHz CPU and Geforce GPU board (RTX2080Ti), which is necessary to execute a deep-learning-based fingertip detection algorithm in real time.

The input image flow was 720×540 pixels at a rate of 500 fps and camera cycle time $\tau_h = 2$ ms. For finger ROI calculation, k_1 and k_2 are set to 1.4 and 0.7, respectively. Assuming that a tapping action can be finished within 0.2 s, N is set to 100 with 500 fps HFR video as input. For peak detection, A_{TH} and T are set to 0.2 pixel/frame and 0.15 s by default, respectively.

(a) Execution time for Finger ROI extraction: The finger ROI extraction algorithm is divided into hand landmark detection and ROI area calculation. The landmark detection network required 32 ms to process an image with 720×540 pixels. Based on the landmarks, one fingertip ROI calculation takes 0.05 ms; the time of ten fingertip ROIs for both hands is 0.5 ms. Thus, the fingertip ROI calculation takes $\tau_d = 32.5$ ms in total, which means that fingertip ROIs are updated at 30 fps.

(b) Execution time of velocity estimation: POC speed is determined mainly by the ROI size. The average block size of the fingertip ROIs is 48×48 pixels. It takes 0.46 ms for POC to estimate the velocities of all finger ROIs individually. This time is less than the camera cycle time of $\tau_h = 2$ ms. Thus, the actual velocity estimation speed is the same as the camera frame rate, that is, 500 fps.

(c) Execution time for tapping detection: In the tapping detection algorithm, bandpass filtering and peak detection are executed every 20 frames, which requires an execution time shorter than 40 ms. For a single fingertip ROI, bandpass filtering takes 0.13 ms and peak detection takes 0.02 ms. The 10-time tapping detection for all fingertips takes 1.5 ms that is significantly shorter than the required 40 ms. Therefore, tapping detection can be performed at 25 fps in real-time.

6.5 Experiments for fingertip velocimeter

6.5.1 A force sensor experiment

To verify the effectiveness of our fingertip velocimeter, we conducted an experiment that compared the force signal collected by the force sensor with the velocity signal measured by our system. As Fig. 6.8 illustrates, the force sensor (PFS055YA251U6, Leptrino, Japan) is fixed on the table and its surface can sense the force information at a sample frequency of 1200 Hz. The high-speed camera is installed at a distance of 50 cm directly over the force sensor and can capture the index finger tapping at 500 fps with

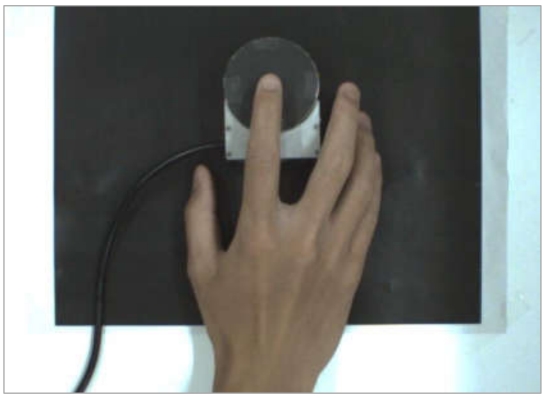


Figure 6.8: The scene of the comparison experiment.

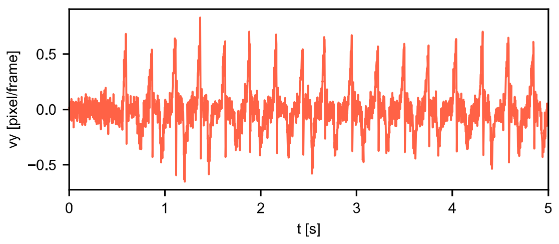
a resolution of 720×540 pixels. The parameters of the algorithms are the same as those stated in Subsection 6.4.

Fig. 6.9 shows (a) vertical velocity signal measured by our system, (b) its frequency distribution, (c) bandpass filtered signal of 20–50 Hz, and (d) force signal collected by the force sensor.

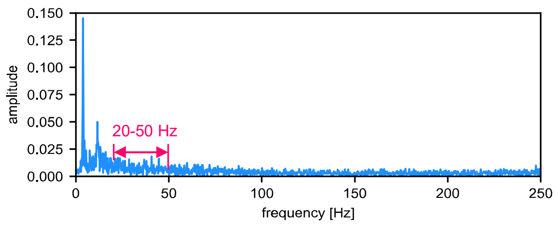
DL-based fingertip detection automatically detects the index finger position to update the fingertip ROIs for POC in real-time. Frame-by-frame POC calculation enables our software-based velocimeter to output a sub-pixel-accuracy velocity signal with a sampling frequency of 500 Hz. The bandpass-filtered signal depicts the tapping moment more intuitively by removing the low-frequency tapping components and high-frequency random noise.

The velocity signal measured by our fingertip velocimeter maintained a high consistency with the force signal collected by the force sensor. Both types of signals clearly showed that the index finger tapped 17 times during the 5 s test. The velocity signals measured by our system can completely record the tapping procedure, whereas the force sensor can only record the moment at which the fingertip contacts the surface.

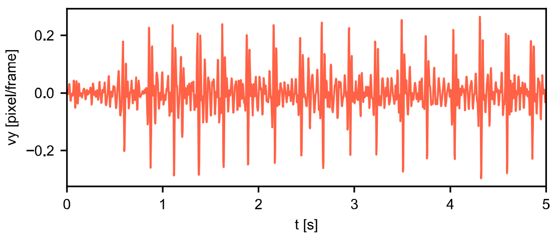
This experiment demonstrates that our mount-free fingertip velocimeter can collect the same high-accuracy velocity signals as a force sensor. Moreover, it keep tracking the



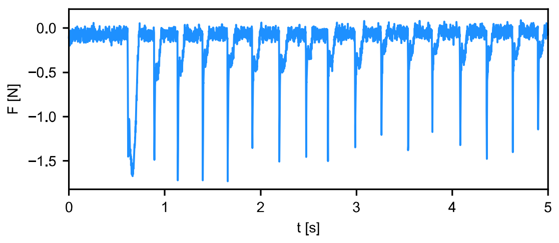
(a) Vertical velocity signal measured by our system.



(b) Frequency distribution of the vertical velocity signal.



(c) Bandpass filtered velocity signal of 20–50 Hz.



(d) Force signal collected by the force sensor.

Figure 6.9: Experimental results of the comparison experiment.

fingertip motion instead of sensing the finger speed at certain moments.

6.5.2 Comparison experiment

In this experiment, we compared our method with two popular classic methods (color-contour [113], depth camera [117]) and two recent AI-based methods (OpenPose

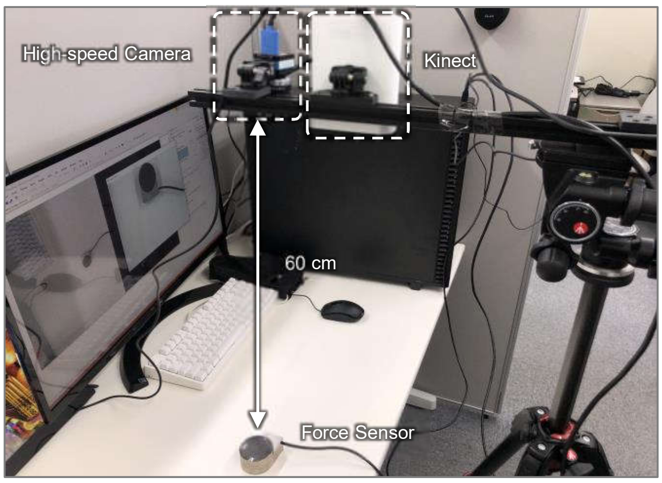


Figure 6.10: Configuration of comparison experiment.

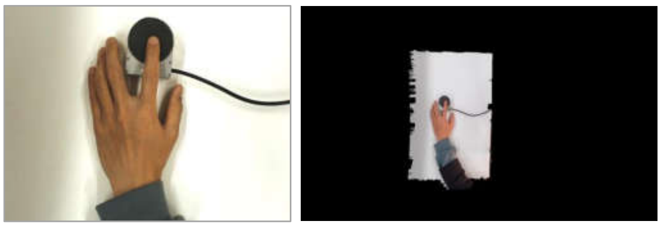


Figure 6.11: Recorded images in comparison experiment.

[123], mediapipe [130]). As Fig. 6.10 shown, a high-speed camera and a Kinect (Azure Kinect, Microsoft, America) are installed 60 cm away from the table plane. The high-speed camera outputs high-speed video (720×540@500fps) as input of our method, the color-contour method and the two AI methods, and the Kinect device outputs real-time RGB video (1280×720@30fps) and depth video (640×576@30fps) for the depth-camera-based method. The experimental images are shown in Fig. 6.11: left is the hand image recorded by the high-speed camera; right is the depth-based-segmented color image captured by Kinect. During the experiment, the index finger of the left hand keeps tapping for 3 seconds with light condition changing from $t = 0.7$ s to $t = 2.4$ s. For fingertip detection methods, velocity in the vertical direction was calculated by the difference of fingertip positions of two adjacent frames. Our method can directly output velocity signal by the POC algorithm with two adjacent frames as input.

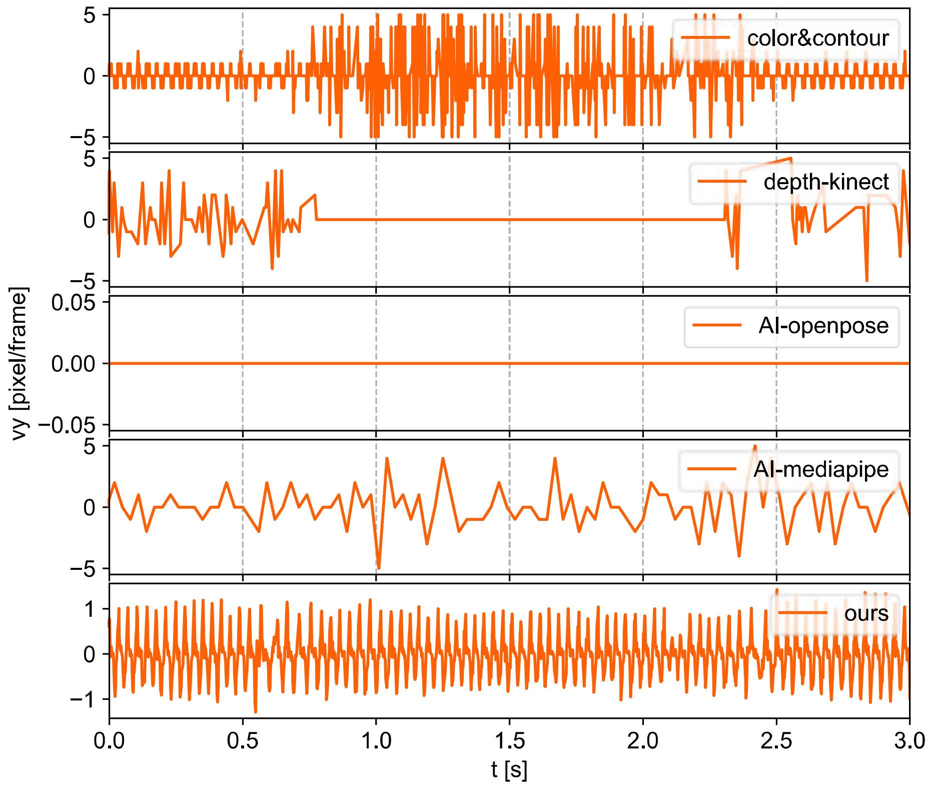


Figure 6.12: Velocity signals of the comparison experiment.

Fig. 6.12 shows the performance of tested methods. Color-contour method can run within 2 ms and output high-frequency velocity signals. Its signal appears clear pattern until t =0.7 s but become messy when the light changes from t = 0.7 s to t = 2.4 s. It means that color-contour-based method can efficiently detect fingertip position but will be easily affected by the light changes. Kinect-based method shows the same trend with lower sampling frequency because it also uses the color and contour information for fingertip recognition after the depth-based segmentation and works only at 30 fps. OpenPose can detect the fingertip positions at 25 fps on GPU and adapt light changes well. Its velocity keeps 0 because its detection accuracy is too rough for accurate fingertip velocity estimation and just outputs a static fingertip position when the fingertip moves with a small motion. Compared to OpenPose, medeiapipe was deeply optimized both from network structure and engineering completion. It can robustly output fingertip detection

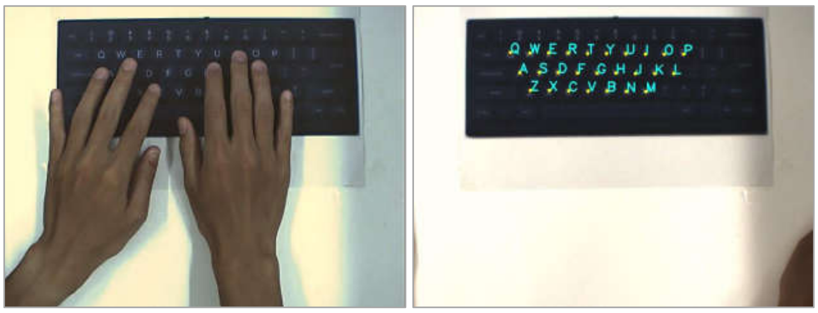


Figure 6.13: Keyboard scene captured by the camera and detected character distribution.

results at 30 fps and achieve higher accuracy than OpenPose. However, its sample frequency is relatively limited by its low sampling frequency at 15 Hz so that it still cannot provide clear patterns for finger tapping. The velocity signal of our method simultaneously meets the requirements of high speed and high robustness. Clear tapping pattern is well displayed, which contains a sharp change when the fingertip contacts a plane.

This experiment demonstrated that our proposed method can address low robustness of traditional color-contour-based methods and low speed of recent popular AI methods. With high-frequency and high-robustness velocity signals, more complicated dynamic analysis of finger motions are enabled.

6.5.3 Virtual keyboard experiment

To verify the effectiveness of our system in simultaneously detecting the tapping actions of multiple fingers, we conducted a virtual keyboard experiment that converts tapping signals into tapped characters to function as an intelligent input device.

As shown in Fig. 6.13, the keyboard is printed on a piece of A4 paper pasted on the table. The two hands can tap on keyboard paper freely similar to their operations on real keyboards. The camera is installed over the table at a distance of 60 cm to monitor hand motions and output HFR image sequence of 720×540 pixels at 500 fps. The keyboard paper can be placed on the table arbitrarily, and character positions on the keyboard

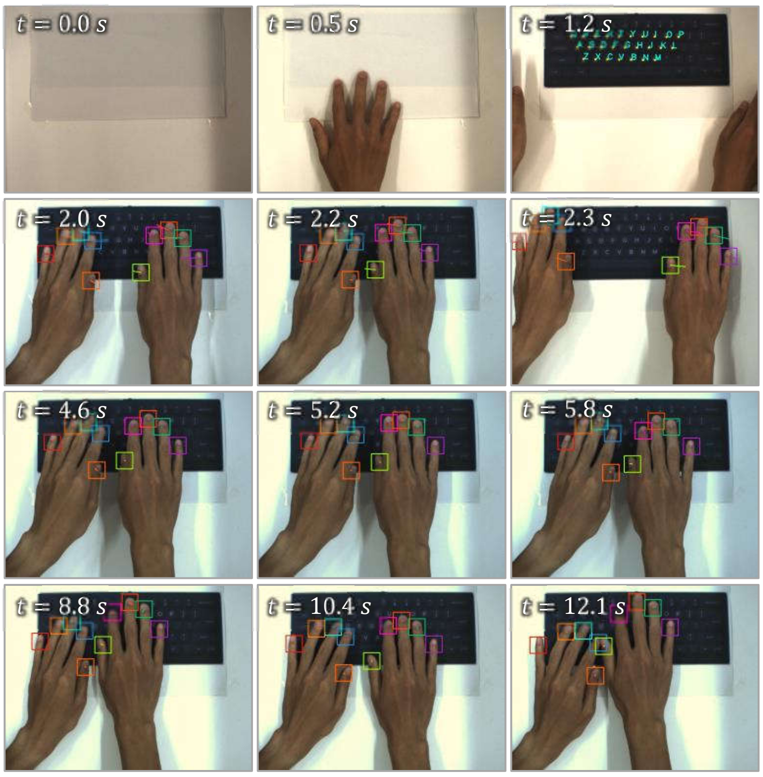


Figure 6.14: Processed image sequence in the virtual keyboard experiment.

can be automatically detected by a character recognition algorithm [132]. With fingertip positions detected by AI-based fingertip detection and character positions detected by the OCR algorithm, we can determine which character a fingertip is above by finding the closest character around the fingertip.

Fig. 6.14 shows processed image sequence of the whole experiment process. The entire experiment was divided into four stages: keyboard detection, hand movement, non-contact tapping, and contact tapping. Ten fingertip ROIs marked in different colors were updated using DL-based fingertip detection. During t = 0–1.5 s, when the keyboard appeared in the camera view, character distribution was detected in real-time. In the third stage, t = 1.8–3.1 s, two hands moved in the horizontal direction quickly. Then, the ten fingertips tapped above the keyboard and did not contact the keyboard during t = 4.2—6.2 s. Finally, contact tapping was executed during t = 7.4—14 s.

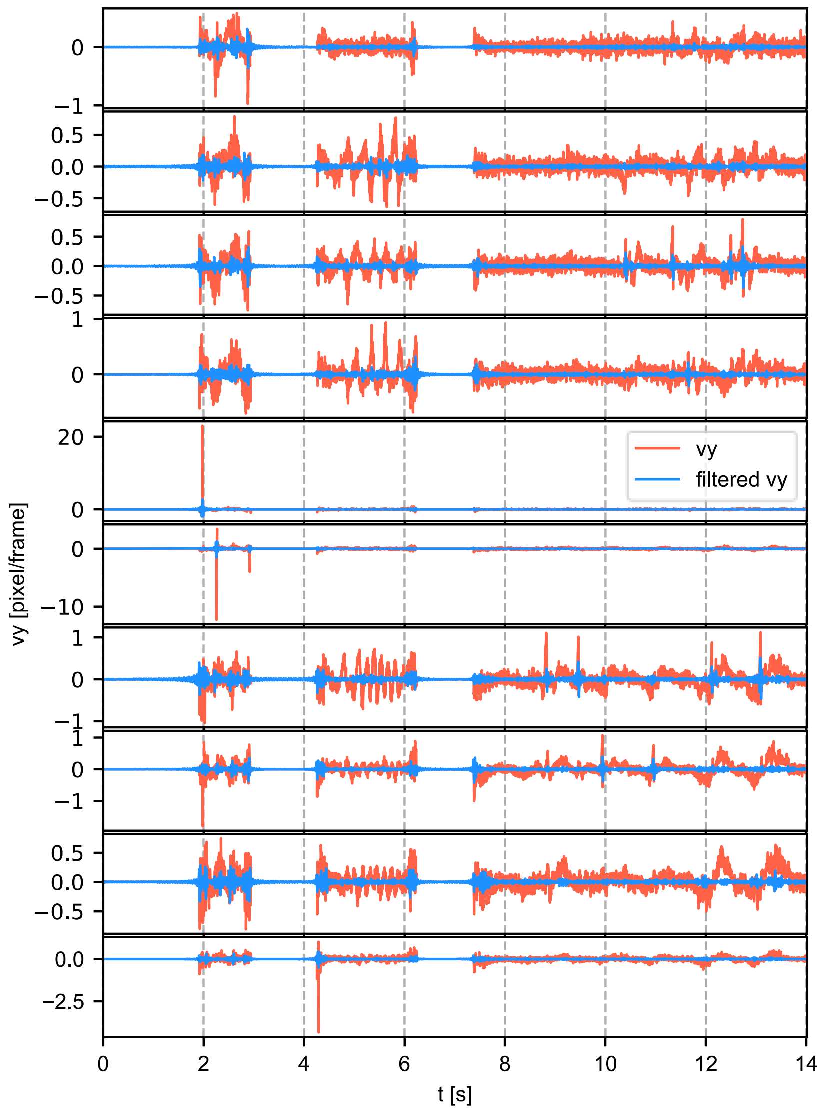


Figure 6.15: Velocity information in the vertical direction of 10 fingertips during the 14 s test. From top to bottom are the thumb, index, middle, ring, and little fingers of the left hand, and those of the right hand.

Fig. 6.15 shows the vertical velocity values of 10 fingertips during the 14 s test. When the two hands were purely moving in the horizontal direction, the velocity signals appeared chaotic owing to the large shifting amplitudes. In the non-contact tapping stage, the original vertical velocities exhibited clear tapping patterns. However, the amplitudes of the filtered velocities decreased significantly because there were no contact actions that could generate obvious frequency components in the range of 20–50 Hz. In the

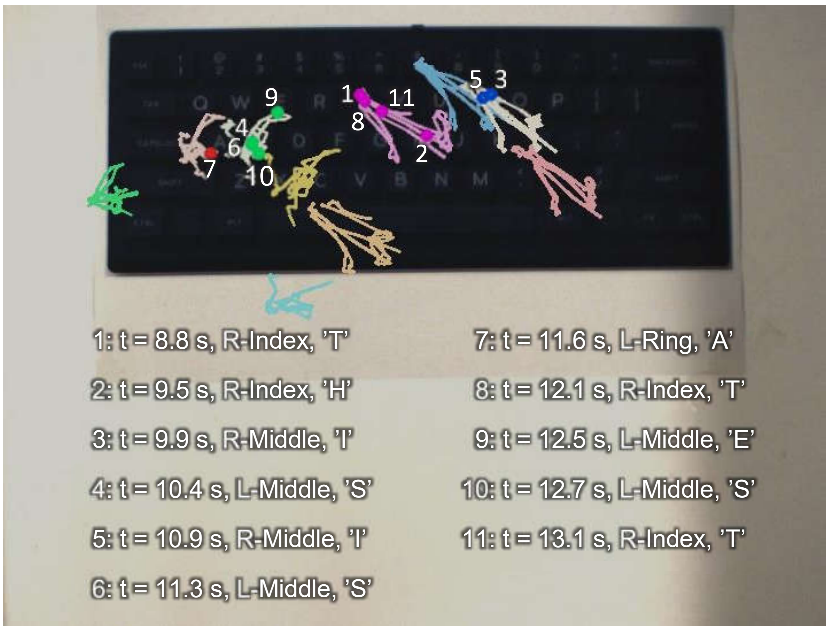

9
1
11
5 3
8
4
6
2
7
10
1: t = 8.8 s, R-Index, 'T'
2: t = 9.5 s, R-Index, 'H'
3: t = 9.9 s, R-Middle, 'I'
4: t = 10.4 s, L-Middle, 'S'
5: t = 10.9 s, R-Middle, 'I'
6: t = 11.3 s, L-Middle, 'S'
7: t = 11.6 s, L-Ring, 'A'
8: t = 12.1 s, R-Index, 'T'
9: t = 12.5 s, L-Middle, 'E'
10: t = 12.7 s, L-Middle, 'S'
11: t = 13.1 s, R-Index, 'T'

Figure 6.16: Trajectories and tapping results of 10 fingertips at the contact-tapping stage.

contact-tapping stage, the velocities before and after filtering both exhibited obvious tapping patterns. This indicates that our FFT-based tapping detection can distinguish contact tapping from invalid non-contact tapping.

Fig. 6.16 shows the trajectories of ten fingertips and their tap results during the contact-tapping stage. DL-based fingertip detection can record all fingertip positions to recognize tapped characters in real time. Tapping detection is effective only when the hand is static because a finger cannot simultaneously move and tap on the table plane. The filtered vertical velocities of the ten fingertips are shown in Fig. 6.17 to present the processing procedure. From the velocity signals, we can clearly see when a character is tapped, and which finger performs the tapping action. Regarding the two most commonly used fingers, the middle finger of the left hand tapped the character 'S' for three times and index finger of the right hand tapped the character 'T' for three times. All characters tapped by the four fingers make up the sentence "THIS IS A TEST".

The virtual keyboard experiment demonstrates that our system can simultaneously

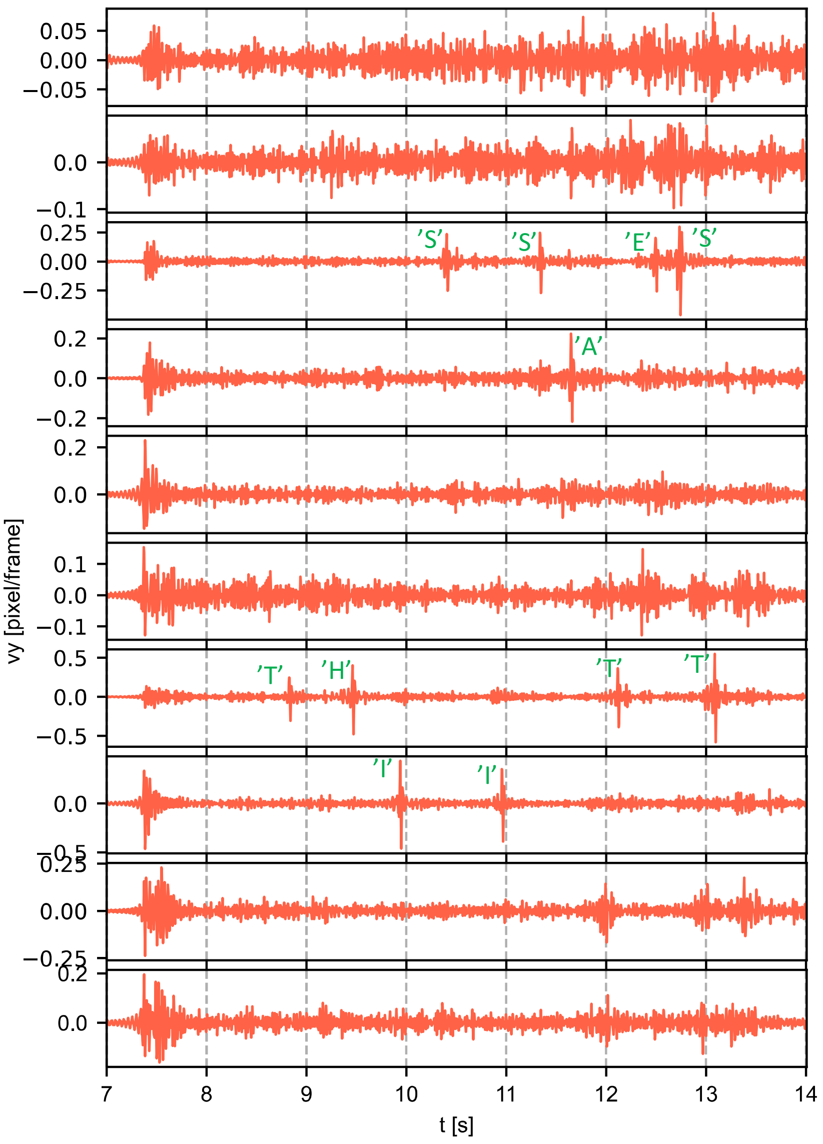


Figure 6.17: Filtered velocities and tapped characters at the contact-tapping stage. From top to bottom are the thumb, index, middle, ring, and little fingers of the left hand, and those of the right hand.

detect the tapping actions of multiple fingers and can effectively distinguish invalid tapping actions. Because any customized plane can be used to collect tapping signals, we believe that our system has broad application prospects in the field of human-computer interactions.

6.6 Concluding remarks

In this study, we propose an HFR-video-based fingertip velocimeter that can output the velocity signals of multiple fingers at 500 fps by combining DL-based fingertip detection with DIC-based velocity estimation. Based on this framework, we built a multi-finger tapping detection system that can simultaneously detect the tapping actions of multiple fingers. To verify its effectiveness, a comparison experiment with the force sensor, as well as a virtual keyboard experiment, was conducted. The experimental results show that our system can output high-accuracy and high-frequency velocity signals obtained from multiple fingers and can detect valid contact-tapping actions in real time.

As DIC-based velocity estimation is easily affected by invalid background information, in future research, we plan to introduce image segmentation to remove background interference for achieving a more accurate velocity estimation. Moreover, we focus on apparent 2D velocity measurement of fingertips in images due to single-camera-based environment. For future works, we will consider 3D velocities of fingertips to know fingertip environment interaction in detail by extending our system to multi-camera stereo one.

Chapter 7

Conclusions

In this study, to realize the dynamic analysis for high-speed scene where the moving actions could not be directly seen by human naked eyes, we proposed a novel concept named HFR-video-based software sensor that can realize real-time and long-term dynamic analysis by combining parallelized DIC algorithms for high-speed sensing with real-time analysis algorithms developed for specific applications.

The working procedure of proposed software sensor consists three steps: HFR-video capturing for recording original light-level image data, high-speed full-field velocity/displacement estimation, and real-time dynamic analysis facing to specific application.

To realize high-speed full-filed image velocity estimation, we developed two kinds of GPU-based DIC algorithms, GPU-based batch POC algorithm and GPU-based multi POC algorithm. GPU-based batch POC algorithm is developed for parallelizing velocity estimation of multiple pair of subimages to realize millisecond-level full-filed velocity field calculation. With the strong Geforce RTX 3090 GPU platform, our proposed batch POC algorithm can estimate full-field velocity/displacement distribution in an image of 1920×1080 pixels in 1 ms at 100 fps. To improve reliability of POC under noisy and weak-texture cases, GPU-based multi POC algorithm was developed to estimate final displacement value by synchronizing the estimated results between test image and multiple reference images generated by interpolation. A series of experiments were conducted to demonstrate that multi POC outperforms traditional POCs under different interferences at a faster speed.

Then we developed three kinds of software sensors facing to vibration visualization, rotation measurement and finger tapping analysis.

Vibration visualization sensor is proposed to help human more intuitively understand vibration distribution that is usually hard to capture. Through executing real-time STFT analysis on the full-field velocity/displacement calculated by GPU-based batch POC algorithm, our proposed software sensor for vibration visualization realized displaying frequency responses in the range of 0–500 Hz on a computer at dozens of frames per second with a HFR-video of 1920×1080 resolution at 1000 fps as input. Real-time vibration visualization experiments including a cymbal-based free vibration and a metal-box-based forced vibration were conducted.

The second rotation sensor can simultaneously measure rotation angles of multiple high-speed rotating targets by real-time angle searching in the similarity values calculated by the GPU-based batch POC algorithm. With 500fps HFR video as input, our software sensor realized rotation angle detection of fast-rotating gears working at 2400 prm.

To apply our software sensor concept in more natural and complicated scene, we developed the finger velocimeter for finger tapping analysis. With HFR video as input, deep-learning-based finger detection is introduced to detect and update positions of multiple fingertips in real time. High-frequency sub-pixel-precision finger velocity information is calculated by operating POC algorithm on every frame. Finally, by extracting the high-frequency components that develops when the fingertip actively contacts something, our finger tapping sensor realized real-time tapping detection of multiple fingers. A series of experiments including comparison experiment with current popular solutions were conducted to demonstrate the advantages of the proposed finger tapping sensor. A virtual keyboard was also presented to show our software sensor can effectively and accurately detect finger tapping of 10 fingers in real time.

The following issues remain to be solved in the future. First, more accurate region selection for DIC calculation still remains a challenge. Unnecessary background information will decrease the accuracy and robustness of DIC estimation. So in the future research, we will try introducing image segmentation to obtain pixel-level target regions. Second, current DIC calculation only focus on 2D-level measurement. However, most real applications shows strong requirement of 3D measurement especially in the fault defection, structure health analysis fields. Thus, we plan to research high-speed 3D DIC algorithm to explore more widely application scenes.

Bibliography

[1] D. Goyal, and B. S. Pabla, “The vibration monitoring methods and signal processing techniques for structural health monitoring: a review,” *Arch. Comput. Methods Eng.*, vol. 23, no. 4, pp. 585-594, 2016.

[2] A. P. Daga, and L. Garibaldi, “Machine vibration monitoring for diagnostics through hypothesis testing,” *Information*, vol. 10, no. 6, 204, 2019.

[3] X. Lei, and Y. Wu, “Research on mechanical vibration monitoring based on wireless sensor network and sparse Bayes,” *EURASIP J. Wireless Commun. Networking*, vol. 2020, *225*, 2020.

[4] Q. Huang, B. Tang, and L. Deng, “Development of high synchronous acquisition accuracy wireless sensor network for machine vibration monitoring,” *Measurement*, vol. 66, pp. 35-44, 2015.

[5] A. Cigada, P. Mazzoleni, and E. Zappa, “Vibration monitoring of multiple bridge points by means of a unique vision-based measuring system,” *Exp. Mech.*, vol. 54, no. 2, pp. 255-271, 2014.

[6] E. Caetano, S. Silva, and J. Bateira, “A vision system for vibration monitoring of civil engineering structures,” *Exp. Tech.*, vol. 35, no. 4, pp. 74-82, 2011.

[7] E. Balms, M. Basseville, F. Bourquin, L. Mevel, H. Nasser, and F. *Treyssede*, “Merging sensor data from multiple temperature scenarios for vibration monitoring of civil structures,” *Struct. Health Monit.*, vol. 7, no. 2, pp. 129-142, 2008.

[8] A. Zona, “Vision-based vibration monitoring of structures and infrastructures: An overview of recent applications,” *Infrastructures*, vol. 6, no. 1, 4, 2021.

[9] L. F. Ramos, R. Aguilar, P. B. Lourenço, and S. Moreira, ”Dynamic structural health monitoring of Saint Torcato church,” *Mech Syst Signal Process*, vol. 35, no. 1-2, pp. 1-15, 2013.

[10] J. Bolhaar, M. Lindeboom, and B. Van Der Klaauw, ”A dynamic analysis of the demand for health insurance and health care,” *Eur Econ Rev*, vol. 56, no. 4, pp. 669-690, 2012.

[11] L. Mariani, F. Pastore, and M. Pezze, ”Dynamic analysis for diagnosing integration faults,” *Trans. Softw. Eng.*, vol. 37, no. 4, pp. 486-508, 2010.

[12] S. Sarkar, K. Mukherjee, S. Sarkar, and A. Ray, ”Symbolic dynamic analysis of transient time series for fault detection in gas turbine engines,” *J Dyn Syst Meas Control*, vol. 135, no. 1, 014506, 2013.

[13] F. L. M. dos Santos, B. Peeters, J. Lau, et al, “The use of strain gauges in vibration-based damage detection,” J. Phys: Conf. Series IOP Publishing, vol. 628, no.1, 012119, 2015.

[14] H. S. Park, H. Y. Lee, S. W. Choi, and Y. Kim, “A practical monitoring system for the structural safety of mega-trusses using wireless vibrating wire strain gauges,” *Sensors*, vol. 13, no. 12, pp. 17346-17361, 2013.

[15] M. Y. Cheng, K. W. Liao, Y. F. Chiu, Y. W. Wu, S. H. Yeh, and T. C. Lin, “Automated mobile vibration measurement and signal analysis for bridge scour prevention and warning,” Automat. Construct., vol. 134, 104063, 2022.

[16] D. C. Kammer, and M. L. Tinker, “Optimal placement of triaxial accelerometers for modal vibration tests,” Mech. Syst. Signal Process., vol. 18, no. 1, pp. 29-41, 2004.

[17] X. Meng, A. H. Dodson, and G. W. Roberts, “Detecting bridge dynamics with GPS and triaxial accelerometers,” Eng. Struct., vol.29, no.11, pp. 3178-3184, 2007.

[18] A. Umeda, M. Onoe, K. Sakata, T. Fukushia, K. Kanari, H. Iioka, and T. Kobayashi, “Calibration of three-axis accelerometers using a three-dimensional vibration generator and three laser interferometers,” Sens. Actuators A: Phys., vol. 114, no. 1, pp. 93-101, 2004.

[19] A. Sabato, C. Niezrecki C, and G. Fortino, “Wireless MEMS-based accelerometer sensor boards for structural vibration monitoring: a review,” IEEE Sens. J., vol. 17, no. 2, pp. 226-235, 2016.

[20] S. Kalaiselvi, L. Sujatha, and R. Sundar, “Fabrication of MEMS accelerometer for vibration sensing in gas turbine,” Proc. 2018 IEEE SENSORS, pp. 1-4, 2018.

[21] X. Zhang, Q. Shen, and X. Liu, “A High Sensitivity MEMS-based Accelerometer with Reduced Cross-axis Coupling for Vibration Detection,” Proc. IEEE Int. Conf. Unmanned Syst., pp. 951-954, 2019.

[22] S. J. Rothberg, M. S. Allen, P. Castellini, D.Di Maio, J. J. J. Dirckx, D. J. Ewins, B. J. Halkon, P. Muyshondt, N. Paone, T. Ryan, H. Steger, E. P. Tomasini, S. Vanlanduit, and J. F. Vignola, “An international review of laser Doppler vibrometry: Making light work of vibration measurement,” Opt. Lasers Eng., vol. 99, pp. 11-22, 2017.

[23] N. Hasheminejad, C. Vuye, J. Dirckx, and S. Vanlanduit, “A comparative study of laser Doppler vibrometers for vibration measurements on pavement materials,” Infrastructures, vol. 3, no. 4, 47, 2018.

[24] H. Khalil, D. Kim, J. Nam, K. Park, “Accuracy and noise analyses of 3D vibration measurements using laser Doppler vibrometer,” Measurement, vol. 94, pp. 883-892, 2016.

[25] H. Nguyen, Z. Wang, P. Jones, and B. Zhao, “3D shape, deformation, and vibration measurements using infrared Kinect sensors and digital image correlation,” Appl. Opt., vol. 56, no. 32, pp. 9030-9037, 2017.

[26] M. Kalybek, M. Bocian, and N. Nikitas, “Performance of optical structural vibration monitoring systems in experimental modal analysis,” Sensors, vol. 21, no. 4, 1239, 2021.

[27] Z. Qiu, X. Wang, X. M. Zhang, and J. Liu, “A novel vibration measurement and active control method for a hinged flexible two-connected piezoelectric plate,” Mech. Syst. Signal Process. vol. 107, pp. 357-395, 2018.

[28] L. Wu, Y. Su, Z. Chen, S. Chen, S. Cheng, and P. Lin, “Six-degree-of-freedom generalized displacements measurement based on binocular vision,” Struct. Contr. Health Monit., vol. 27, no. 2, e2458, 2020.

[29] J. Luo, B. Liu, P. Yang, X. Fan, “High-speed vision measurement of vibration based on an improved ZNSSD template matching algorithm,” Syst. Sci. Contr. Eng., vol. 10, no. 1, pp. 43-54, 2022.

[30] Y. Wang, J. Brownjohn, J. A. Jimenez Capilla, K. Dai, W. Lu and K. Y. Koo, “Vibration investigation for telecom structures with smartphone camera: case studies,” J. Civil Struct. Health Monit., vol. 11, no. 3, pp. 757-766, 2021.

[31] D. H. Diamond, P. S. Heyns, and A. J. Oberholster, “Accuracy evaluation of subpixel structural vibration measurements through optical flow analysis of a video sequence,” Measurement, vol. 95, pp. 166-172, 2017.

[32] C. Z. Dong, O. Celik, F. N. Catbas, E. J. O’Brien and S. Taylor, “Structural displacement monitoring using deep learning-based full field optical flow methods,” Struct. Infrastruct. Eng., vol. 16, No. 1, pp. 51-71, 2020.

[33] T. J. Beberniss, and D. A. Ehrhardt, "High-speed 3D digital image correlation vibration measurement: Recent advancements and noted limitations," Mech. Syst. Signal Process., vol. 86, pp. 35-48, 2017.

[34] P. L. Reu, D. P. Rohe, and L. D. Jacobs, "Comparison of DIC and LDV for practical vibration and modal measurements," Mech Syst. Signal Process., vol. 86, pp. 2-16, 2017.

[35] M. Jiang, Q. Gu, T. Aoyama, T. Takaki, and I. Ishii, "Real-time vibration source tracking using high-speed vision," IEEE Sens. J., vol. 17, no. 5, pp. 1513-1527, 2017.

[36] K. Shimasaki, T. Okamura, M. Jiang, T. Takaki, and I. Ishii, "Real-time high-speed vision-based vibration spectrum imaging," Proc. IEEE/SICE Int. Symp. Syst. Integr., pp. 474-477, 2019.

[37] Q. Hu, S. He, S. Wang, Y. Liu, Z. Zhang, L. He, F. Wang, Q. Cai, R. Shi, and Y. Yang, "A high-speed target-free vision-based sensor for bus rapid transit viaduct vibration measurements using CMT and ORB algorithms," Sensors, vol.17, no.6, 1305, 2017.

[38] B. Pan, K. Qian, H. Xie, and A. Asundi, "Two-dimensional digital image correlation for in-plane displacement and strain measurement: a review," *Meas. Sci. Tech.*, vol. 20, no. 6, *062001*, 2009.

[39] G. Y. Jeong, A. Zink-Sharp, and D. P. Hindman, "Tensile properties of earlywood and latewood from loblolly pine (Pinus taeda) using digital image correlation," *Wood Fiber Sci,*, vol. 41, no. 1, pp. 51-63, 2009.

[40] R. Fedele, B. Raka, F. Hild, and S. Raux, "Identification of adhesive properties in

GLARE assemblies using digital image correlation," *J. Mech. Phys. Solids*, vol. 57, no. 7, pp. 1003-1016, 2009.

[41] S. Yoneyama, and H. Ueda, "Bridge deflection measurement using digital image correlation with camera movement correction," *Mater. Trans.*, vol. 53, no. 2, pp. 285-290, 2012.

[42] W. H. Peters, and W. F. Ranson W F, "Digital imaging techniques in experimental stress analysis," *Opt. Eng.*, vol. 21, no. 3, pp. 427-431, 1982.

[43] A. Giachetti, "Matching techniques to compute image motion," *Image Vision Comput.*, vol. 18, no. 3, pp. 247-260, 2000.

[44] B. Pan, Z. Wang, and H. Xie, "Generalized spatial-gradient-based digital image correlation for displacement and shape measurement with subpixel accuracy," *J. Strain Anal. Eng. Des.*, vol. 44, no. 8, pp.659-669, 2009.

[45] W. Tong, "An evaluation of digital image correlation criteria for strain mapping applications," Strain, vol. 41, no. 4, pp. 167-175, 2005.

[46] B. Pan, "Recent progress in digital image correlation," *Exp. Mech.*, vol.51, no. 7, pp. 1223-1235, 2011.

[47] M. A. Sutton, C. Mingqi, W. H. Peters, Y. J. Chao, and S. R. McNeill, "Application of an optimized digital correlation method to planar deformation analysis," *Image Vision Comput.*, vol. 4, no. 3, pp. 143-150, 1986.

[48] D. J. Chen, F. P. Chiang, Y. S. Tan, and H. S. Don, "Digital speckle-displacement measurement using a complex spectrum method," *Appl. Opt.*, vol. 32, no. 11, pp. 1839-1849, 1993.

[49] H. W. Schreier, J. R. Braasch, and M. A. Sutton, "Systematic errors in digital image correlation caused by intensity interpolation," *Opt. Eng.*, vol. 39, no. 11, pp. 2915-2921, 2000.

[50] J. Zhang, G. Jin, S. Ma, and L. Meng, "Application of an improved subpixel registration algorithm on digital speckle correlation measurement," *Opt. Laser Tech.*, vol. 35, no. 7, pp. 533-542, 2003.

[51] H. Jin, and H. A. Bruck, "Pointwise digital image correlation using genetic algorithms," *Exp. Tech.*, vol. 29, no. 1, pp. 36-39, 2005.

[52] B. Pan, K. Li, and W. Tong, "Fast, robust and accurate digital image correlation calculation without redundant computations," *Exp. Mech.*, vol. 53, no. 7, pp. 1277-1289, 2013.

[53] L. Luu, Z. Wang, M. Vo, T. Hoang and J. Ma, "Accuracy enhancement of digital image correlation with B-spline interpolation," *Opt. Lett.*, vol.36, no. 16, pp. 3070-3072, 2011.

[54] M. Ren, J. Liang, Z. Tang, X. Guo and L. G. Li, "Optimized interpolation filter for digital image correlation methods," *J. Xian Jiaotong Univ.*, pp. 65-70, 2014.

[55] M. Ren, J. Liang, and B. Wei, "Accurate B-spline-based 3-D interpolation scheme for digital volume correlation," *Rev. Sci. Instr.*, vol. 87, no. 12, *125114*, 2016.

[56] Y. Zhou, C. Sun, and J. Chen, "Adaptive subset offset for systematic error reduction in incremental digital image correlation," *Opt. Lasers Eng.*, vol. 55, pp. 5-11, 2014.

[57] D. Wang, Y. Jiang, W. Wang, and Y. Wang, "Bias reduction in sub-pixel image registration based on the anti-symmetric feature," *Meas. Sci. Tech.*, vol. 27, no. 3, *035206*, 2016.

[58] B. Pan, “Bias error reduction of digital image correlation using Gaussian pre-filtering,” *Opt. Lasers Eng.*, vol. 51, no. 10, pp. 1161-1167, 2013.

[59] P. L. Reu, W. Sweatt, T. Miller, and D. Fleming, “Camera system resolution and its influence on digital image correlation,“ *Exp. Mech.*, vol. 55, no. 1, pp. 9-25, 2015.

[60] H. W. Schreier, J. R. Braasch, and M. A. Sutton, “Systematic errors in digital image correlation caused by intensity interpolation,“ *Opt. Eng.*, vol. 39, no. 11, pp. 2915-2921, 2000.

[61] J. M. Vassoler, and E. A. Fancello, “Error analysis of the digital image correlation method,“ *Mecánica Computacional*, vol. 29, no. 61, pp. 6149-6161, 2010.

[62] I. Ishii, T. Tatebe, Q. Gu, Y. Moriue, T. Takaki and K. Tajima, “2000 fps real-time vision system with high-frame-rate video recording,” *Proc. IEEE Int. Conf. Robot. Automat.*, pp.1536-1541, 2010.

[63] M. Hirabayashi, Y. Saito, K. Murakami, A. Ohsato, S. Kato and M. Edahiro, “Vision-Based Sensing Systems for Autonomous Driving: Centralized or Decentralized?,” *J. Robot. Mechatron.*, vol. 33, no. 3, pp. 686-697, 2021.

[64] Y. Nie, T. Takaki, I. Ishii, and H. Matsuda, “Algorithm for automatic behavior quantification of laboratory mice using high-frame-rate videos,” *SICE J. Contr. Meas. Syst. Integr.*, vol. 4, no. 5, pp. 322-331, 2011.

[65] Y. Yoshimoto, and H. Tamukoh, “FPGA Implementation of a Binarized Dual Stream Convolutional Neural Network for Service Robots,” *J. Robot. Mechatron.*, vol. 33, no. 2, pp. 386-399, 2021.

[66] X. Jiang, Q. Gu, T. Aoyama, T. Takaki, and I. Ishii, “A High-Speed Vision System with Multithread Automatic Exposure Control for High-Dynamic-Range Imaging,” *J. Robot. Mechatron.*, vol. 30, no. 1, pp. 117-127, 2018.

[67] J. Takei, S. Kagami, and K. Hashimoto K, "3,000-fps 3-D shape measurement using a high-speed camera-projector system," *Proc. IEEE/RSJ Int. Conf. Intelli. Robot. Syst.*, pp.3211-3216, 2007.

[68] Y. Watanabe, T. Komuro, S. Kagami, and M. Ishikawa, "Real-time visual measurements using high-speed vision," *Proc. Mach. Vision Optomechatr, Appl.* pp.234-242, 2004.

[69] I. Ishii, T. Ichida, Q. Gu, and T. Takaki, "500-fps face tracking system," *J. Real-time Image Process.*, vol. 8, no. 4, pp. 379-388, 2013.

[70] Q. Gu, T. Takaki, and I. Ishii, "Fast FPGA-based multiobject feature extraction," *IEEE Trans. Circ. Syst. Video Tech.*, vol. 23, no. 1, pp.30-45, 2012.

[71] I. Ishii, T. Tatebe, Q. Gu, and T. Takaki, "Color-histogram-based tracking at 2000 fps," *J. Elecr. Imaging* vol. 21, no. 1, *013010*, 2012.

[72] I. Ishii, T. Taniguchi, K. Yamamoto, and T. Takaki, "High-framerate optical flow system," *IEEE Trans. Circ. Syst. Video Tech.*, vol. 22, no. 1, pp. 105-112, 2012.

[73] M. Jiang, T. Aoyama, T. Takaki, and I. Ishii, "Pixel-level and robust vibration source sensing in high-frame-rate video analysis," Sensors, vol. 16, no. 11, *1842*, 2016.

[74] M. Jiang, Q. Gu, T. Aoyama, T. Takaki, and I. Ishii, "Real-time vibration source tracking using high-speed vision," *IEEE Sens. J.*, vol. 17, no. 5, pp. 1513-1527, 2017.

[75] K. Shimasaki, M. Jiang, T. Takaki, I. Ishii and K. Yamamoto, "HFR-video-based honeybee activity sensing," *IEEE Sens. J.*, vol. 20, no. 10, pp. 5575-5587, 2020.

[76] K. Shimasaki, N. Fujiwara, S. Hu, T. Senoo and I. Ishii, "High-frame-rate Video-based Multicopter Tracking System Using Pixel-level Short-time Fourier Transform," *J. Intelli. Robot. Syst.*, vol. 103, no. 2, *36*, 2021.

[77] C. D. Kuglin and D. C. Hines, "The phase correlation image alignment method," *Proc. Int. Conf. Cybern. Soc.*, pp. 163-165, 1975.

[78] H. W. Schreier, J. R. Braasch, and M. A. Sutton, "Systematic errors in digital image correlation caused by intensity interpolation,“ *Opt. Eng.*, vol. 39, no. 11, pp. 2915-2921, 2000.

[79] D. Sehrawat, and N. S. Gill, "Smart sensors: Analysis of different types of IoT sensors," *Proc. Int. Conf. Trends in Electr. Info*, pp. 523-528, 2019.

[80] Y. Watanabe, T. Komuro, S. Kagami, and M. Ishikawa, "Real-timevisual measurements using high-speed vision," *Proc. Mach. Vision Optomechatr, Appl.*, pp. 234-242, 2004.

[81] E. Caetano, S. Silva, and J. A. Bateira, "Vision system for vibration monitoring of civil engineering structures," *Experimental Techniques*, vol. 35, no. 4, pp. 74-82, 2011.

[82] H. G. Maas, and U. Hampel, "Photogrammetric techniques in civil engineering material testing and structure monitoring," *Photogram. Eng. Remote Sens.*, vol. 72, no. 1, pp. 39-45, 2006.

[83] J. G. Chen, N. Wadhwa, F. Durand, T. William, and B. Oral, "Developments with motion magnification for structural modal identification through camera video," *Dyn. Civil Struct.*, vol. 2, pp. 49-57, 2015.

[84] I. Ishii, T. Tatebe, Q. Y. Gu, Y. Moriue, T. Takaki, and K. Tajima, "2000 fps Real-time Vision System with Highframe-rate Video Recording," *Proc. IEEE Int. Conf. Robot. Automat.*, pp. 1536-1541, 2010.

[85] A. Sharma, K. Shimasaki, Q. Y. Gu, J. Chen, T. Aoyama, T. Takaki, I. Ishii, K. Tamura, and K. Tajima, "Super high-speed vision platform for processing 1024$\times$

1024 images in real time at 12500 fps," *Proc. IEEE/SICE Int. Symp. Syst. Integr.*, pp. 544-549, 2016.

[86] Y T. Yamazaki, et al., "A 1ms High-Speed Vision Chip with 3D-Stacked 140GOPS Column-Parallel PEs for Spatio-Temporal Image Processing," *Tech. Dig IEEE Solid State Circuits Conf.*, pp. 82-83, 2017.

[87] I. Ishii, T. Taniguchi, K. Yamamoto, and T. Takaki, "High-frame-rate Optical Flow System," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 1, pp. 105-112, 2012.

[88] I. Ishii, T. Ichida, Q. Y. Gu, and T. Takaki, "500-fps Face Tracking System," *J. Real-time Image Process*, vol. 8, no. 4, pp. 379-388, 2013.

[89] Y. Nie, I. Ishii, K. Yamamoto, K. Orito, and H. Matsuda, "Real-time scratching behavior quantification system for laboratory mice using high-speed vision," *J. Real Time Image Process.*, vol. 4, no. 2, pp. 181-190, 2009.

[90] K. Shimasaki, N. Fujiwara, S. P. Hu, T. Senoo, and I. Ishii, "High-frame-rate Video-based Multicopter Tracking System Using Pixel-level Short-time Fourier Transform," *J. Intelli. Robot. Syst.*, vol. 103, no. 2, pp. 1-24, 2021.

[91] H. S. Yeo, B. G. Lee, and H. Lim, "Hand tracking and gesture recognition system for human-computer interaction using low-cost hardware," *Multimedia Tools and Applications*, vol. 74, no. 8, pp. 2687-2715, 2015.

[92] L. I. Yang, J. Huang, T. Feng, H. A. Wang, and G. Z. Dai, "Gesture interaction in virtual reality," *Virtual Reality and Intelligent Hardware*, vol. 1, no. 1, pp. 84-112, 2019.

[93] Y. Lee, M. Kim, Y. Lee, J. Kwon, Y. Park and D. L, "Wearable finger tracking and

cutaneous haptic interface with soft sensors for multi-fingered virtual manipulation," *IEEE/ASME Transactions on Mechatronics*, vol. 24, no. 1, pp. 67-77, 2018.

[94] J. Lee, M. Sinclair, M. Gonzalez-Franco, E. Ofek and C. Holz, "TORC: A virtual reality controller for in-hand high-dexterity finger interaction," *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pp. 1-13, 2019.

[95] Z. Lv, A. Halawani, S. Feng, et al, "Touch-less interactive augmented reality game on vision-based wearable device," *Personal and Ubiquitous Computing*, vol. 19, no. 3, pp. 551-567, 2015.

[96] L. Feng, G. W. Ng, and L. Ma, "A Review of An Interactive Augmented Reality Customization Clothing System Using Finger Tracking Techniques as Input Device," *Computational Science and Technology*, pp. 457-467, 2020.

[97] L. Meli, C. Pacchierotti, G. Salvietti, et al, "Combining wearable finger haptics and augmented reality: User evaluation using an external camera and the microsoft hololens," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4297-4304, 2018.

[98] E. Coronado, J. Villalobos, B. Bruno, and F. Mastrogiovanni, "Gesture-based robot control: Design challenges and evaluation with humans," *2017 IEEE international conference on robotics and automation (ICRA)*, pp. 2761-2767, 2017.

[99] S. Chen, H. Ma, C. Yang and M. Fu, "Hand gesture based robot control system using leap motion," *International Conference on Intelligent Robotics and Applications*, pp. 581-591, 2015.

[100] O. Mazhar, B. Navarro, S. Ramdani, et al, "A real-time human-robot interaction framework with robust background invariant hand gesture detection," *Robotics and Computer-Integrated Manufacturing*, vol. 60, pp. 34-48, 2019.

[101] S. Kshirsagar, S. Sachdev, N. Singh, A. Tiwari, and S. Sahu, ”IoT Enabled Gesture-Controlled Home Automation for Disabled and Elderly,” *2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC)*, pp. 821-826, 2020.

[102] M. Patel, S. Pandya, and S. Patel, ”Hand Gesture based Home Control Device using IoT,” *International Journal of Advanced Research in Computer Science*, vol. 8, no. 5, 2017.

[103] V. Sideridis, A. Zacharakis, G. Tzagkarakis, and M. Papadopouli, ”GestureKeeper: gesture recognition for controlling devices in IoT environments,” *2019 27th European Signal Processing Conference (EUSIPCO)*, pp. 1-5, 2019.

[104] Y. Gu, C. Yu, Z. Li, et al, ”Accurate and low-latency sensing of touch contact on any surface with finger-worn imu sensor,” *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*, pp. 1059-1070, 2019.

[105] M. Hazman, I. N. A. M. Nordin, F. H. M. Noh, et al, ”IMU sensor-based data glove for finger joint measurement,” *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 20, no. 1, pp. 82-88, 2020.

[106] S. Lemak, V. Chertopolokhov, I. Uvarov, et al, ”Inertial sensor based solution for finger motion tracking,” *Computers*, vol. 9, np.2 , pp.40, 2020.

[107] J. McIntosh, P. Strohmeier, J. Knibbe, S. Boring and K. Hornbak, ”Magnetips: Combining fingertip tracking and haptic feedback for around-device interaction,” *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pp. 1-12, 2019.

[108] Y. Ma, Z. H. Mao, W. Jia, C. Li, J. Yang and M. Sun, ”Magnetic hand tracking

for human-computer interface," *IEEE Transactions on Magnetics*, vol. 47, np. 5, pp. 970-973, 2011.

[109] Z. Yang, S. Yan, B. J. F. van Beijnum, B. Li, and P. H. Veltink, "Hand-finger pose estimation using inertial sensors, magnetic sensors and a magnet," *IEEE Sensors Journal*, 2021.

[110] P. Bellitti, A. De Angelis, M. Dionigi, et al, "A wearable and wirelessly powered system for multiple finger tracking," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, np. 5, pp. 2542-2551, 2020.

[111] L. Sbernini, A. Pallotti, and G. Saggio, "Evaluation of a Stretch Sensor for its inedited application in tracking hand finger movements," *2016 IEEE International Symposium on Medical Measurements and Applications (MeMeA)*, pp. 1-6, 2016.

[112] G. Simion, V. Gui, and M. Otesteanu, "Finger detection based on hand contour and colour information," *2011 6th IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI)*, pp. 97-100, 2011.

[113] D. Lee, and S. G. Lee, "Vision-Based Finger Action Recognition by Angle Detection and Contour Analysis," *ETRI journal*, vol. 33, no. 3, pp. 415-422, 2011.

[114] R. M. Gurav, and P. K. Kadbe, "Real time finger tracking and contour detection for gesture recognition using OpenCV," *2015 International Conference on Industrial Instrumentation and Control (ICIC)*, pp. 974-977, 2015.

[115] A. Kulshreshth, C. Zorn, and J. J. LaViola, "Poster: Real-time markerless kinect based finger tracking and hand gesture recognition for HCI," *2013 IEEE Symposium on 3D User Interfaces (3DUI)*, pp. 187-188, 2013.

[116] A. Lekova, D. Ryan, and R. Davidrajuh, "Fingers and gesture recognition with

kinect v2 sensor," *Information Technologies and Control*, vol. 14, np. 3, pp. 24-30, 2016.

[117] X. Ma, and J. Peng, "Kinect sensor-based long-distance hand gesture recognition and fingertip detection with depth information," *Journal of Sensors*, vol. 2018, pp. 1-9, 2018.

[118] G. Ponraj and H. Ren, "Sensor fusion of leap motion controller and flex sensors using Kalman filter for human finger tracking," *IEEE Sensors Journal*, vol. 18, no. 5, pp. 2042-2049, 2018.

[119] D. S. Tran, N. H. Ho, H. J. Yang, S. H. Kim, and G. S. Lee, "Real-time virtual mouse system using RGB-D images and fingertip detection," *Multimedia Tools and Applications*, vol. 80, no. 7, pp. 10473-10490, 2021.

[120] M. Oudah, A. Al-Naji, and J. Chahl, "Hand gesture recognition based on computer vision: a review of techniques," *journal of Imaging*, vol. 6, no. 8, pp. 73, 2020.

[121] S. Lee, M. Choi, H. Choi, M. S. Park, and S. Yoon, "FingerNet: Deep learning-based robust finger joint detection from radiographs," *2015 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pp. 1-4, 2015.

[122] Y. Li, X. Wang, W. Liu, and B. Feng, "Pose anchor: A single-stage hand keypoint detection network," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 7, pp. 2104-2113, 2019.

[123] T. Simon, H. Joo, I. Matthews, and Y. Sheikh, "Hand keypoint detection in single images using multiview bootstrapping," *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 1145-1153, 2017.

[124] C. Raghavachari and G. A. S. Sundaram, "Deep Learning Framework for Finger-

spelling System using CNN," *2020 International Conference on Communication and Signal Processing (ICCSP)*, pp. 469-473, 2020.

[125] C. Barut, E. Kiziltan, E. Gelir, and F. Kiktürk, "Advanced analysis of finger-tapping performance: a preliminary study," *Balkan medical journal*, vol. 30, no. 2, pp. 167, 2013.

[126] S. Huang, N. Bergstrom, Y. Yamakawa, T. Senoo and M. Ishikawa, "Applying high-speed vision sensing to an industrial robot for high-performance position regulation under uncertainties," *Sensors*, vol. 16, no. 8, pp. 1195, 2016.

[127] M. Jiang, Q. Gu, T. Aoyama, T. Takaki, and I. Ishii, "Real-time vibration source tracking using high-speed vision," *IEEE Sensors Journal*, vol. 17, no. 5, pp. 1513-1527, 2017.

[128] A. Garcia, E. Mattison, and K. Ghose, "High-speed vision-based autonomous indoor navigation of a quadcopter," *2015 international conference on unmanned aircraft systems (ICUAS)*, pp. 338-347, 2015.

[129] F. Hild and Roux S, "Digital image correlation," *Optical methods for solid mechanics. A full-field approach*, vol. 367, pp. 183-228, 2012.

[130] F. Zhang, V. Bazarevsky, A. Vakunov, A. Tkachenka, G. Sung, C. L. Chang, and M. Grundmann, "Mediapipe hands: On-device real-time hand tracking," arXiv preprint arXiv:2006.10214, 2020.

[131] H. Foroosh, J. B. Zerubia, and M. Berthod, "Extension of phase correlation to subpixel registration," *IEEE transactions on image processing*, vol. 11, no. 3, pp. 188-200, 2002.

[132] R. Smith, "An overview of the Tesseract OCR engine," *IEEE Ninth international*

conference on document analysis and recognition (ICDAR 2007), vol. 2, pp. 629-633, 2007.

Acknowledgment

First of all, I would like to express my deep thank to my advisor, **Prof. Idaku Ishii**. Ishii Sensei gave me comprehensive advice to guide me grow up step by step from a new researcher. I learned a lot from every-time meeting with Ishii Sensei and I am always deeply shocked by his wide knowledge and deep thinking. He can always draw the kernel point at the first time and provide his precious advice that shows excellent research and education level. Ishii Sensei's research philosophy helps me rebuild my recognition of what is meaningful research and how to promote the research step by step. Moreover, Ishii Sensei is always so kindly to me and provided me much help in my living. I am always grateful to Ishii Sensei and it is my luck to experience my PHD time with his guidance.

Then I would like to thank to **Dr. Kohei Shimasaki** and **Dr. Shaopeng Hu**, who helped me finish my experiments when I had to do remote experiments. Thanks to **Dr. Qing Li**, **Dr. Mengjuan Chen**, **Dr. Wenxiang Qin**, **Dr. Hongyu Dong**, **Dr. Abudoureheman TUNIYAZI** , **Dr. Aliansyah Muhammad ZULHAJ** who always accompany with me and bring me much happiness in my life.

I am also very grateful to **Ms. Yukari Kaneyuki** and **Ms. Michiko Kanzaki**. You helped me finish many troubles about the procedures during my PHD time. Many things are complicated, but you always keep kindly and patient and help me again and again. I will always remember your smile in my heart.

I want to express my sincere thanks to the bachelor and master students in Robotics

Laboratory. They provide me more connections not only in the research but also the life.

I am grateful to the staff in Hiroshima University and the Japanese government who provided me the scholarship and environment which are very important for my research and life in Japan.

I would like to thank my parents, you always support me silently behind me and encourage me to study without interference.

I want to thank my wife **Lin Chen**, I really hope we can start a new stage of happy life together.

Finally, I sincerely wish all the people who I love and all the people who love me keep healthy and happy forever.

May, 2023

Wang Feiyue