university of groningen / faculty of science and engineering

INTEGRATION PROJECT

FINAL REPORT

Non-Contact Vision Based Structural Vibration Measurements

A Comparative Analysis on Tracker Algorithms for Structural Health Monitoring

Author:
Huib van der Veen (S4017021)

First Supervisor:
dr. L. (Liangliang) Cheng
Second Supervisor:
dr. ing. H. (Harm) Kloosterman
Daily Supervisor:
K. (Kun) Xie

Groningen, June 14, 2024

BSc. Industrial Engineering and Management
Faculty of Science and Engineering
University of Groningen

Abstract

This research evaluates the efficiency of various object-tracking algorithms compared to traditional accelerometer measurements in non-contact, vision-based structural vibration analysis. Specifically, the study focuses on validating the use of a smartphone for monitoring vibrations on a cantilever beam, assessing data quality in both time and frequency domains. The study explores the performance of different tracking algorithms, including the CSRT, MIL, KCF, and OF trackers, in capturing structural vibrations. Findings reveal that the OF tracker shows promise due to its high accuracy and sensitivity, as evidenced by its excellent signal-to-noise ratio and precise identification of primary frequency components. A real-world application on a lamppost further explores the potential of the OF tracker in practical scenarios. These findings highlight the potential of vision-based monitoring systems as valuable tools for structural health monitoring.

Keywords: Vision-based monitoring, Structural Health Monitoring (SHM), Object-tracking algorithms, Structural vibration analysis, Non-contact measurement, Smartphone-based monitoring, Signal-to-noise ratio (SNR), Frequency domain analysis, Cantilever beam, Optical Flow (OF) tracker, Accelerometer comparison, Real-world application.

Table of Contents

I. LIST OF ABBREVIATIONS

- CSRT: Channel and Spatial Reliability Tracking
- FFT: Fast Fourier Transform
- fps: Frames Per Second
- KCF: Kernelized Correlation Filter
- KPIs: Key Performance Indicators
- MATLAB: Matrix Laboratory
- MIL: Multiple Instance Learning
- OF: Optical Flow
- OpenCV: Open Source Computer Vision
- RUG: Rijksuniversiteit Groningen
- SHM: Structural Health Monitoring
- SNR: Signal-to-Noise Ratio

II. INTRODUCTION

Vision-based technologies offer an opportunity for non-contact measurement of structural vibration. These technologies can ease structural health monitoring (SHM), enabling more frequent and widespread evaluations at a fraction of the cost and complexity (Feng & Feng, 2018) (Ferraris et al., 2023).

A. *State of the Art*

Traditional single point-based vibration measurement methods have several limitations that affect their effectiveness in SHM. These methods often require direct contact with the machine or structure, leading to potential failures and downtime. Moreover, point measurements can provide data from limited locations, limiting analysis to specific points and potentially overlooking broader vibration problems throughout the machine or structure (Tiboni et al., 2022). These limitations highlight the need for advanced non-contact techniques, such as camera-based measurement techniques (Zhuang et al., 2022) (Feng & Feng, 2018).

The evolution of computer vision has significantly influenced and accelerated recent advancements in SHM. These technologies enable non-contact, efficient monitoring of infrastructures, offering a leap forward from traditional methods. This shift enhances the capability for comprehensive assessments and ensures more accessible, cost-effective, and widespread evaluations (Ferraris et al., 2023).

A contribution to this is a comparative study of vision camera-based vibration analysis with the laser vibrometer method (Muralidharan & Yanamadala, 2021). Their research delves into the application of advanced image processing to detect and analyze structural vibrations, demonstrating the potential of camera-based systems as a cost-effective and accessible option for vibration measurement.

Complementing this, the advancement in the field with a focus on practical applications of camera-based vibration measurement (Baqersad & Di Maio, 2023). Their volume on Computer Vision & Laser Vibrometry highlights comparative studies which assesses the effectiveness of these technologies in SHM of various infrastructures. This collective body of work confirms the validity of vision-based methods and paves the way for broader adoption of vision-based methods in structural dynamics and SHM systems, leading to safer and more efficiently maintained structures.

B. *Contribution*

This paper aims to validate the effectiveness of vision-based monitoring against accelerometer measurements using a smartphone to record the vibrations on a cantilever beam. It assesses the data quality in both the time and frequency domain. Comparing different object-tracking algorithms can help refine these methods in SHM by making vision-based monitoring more accessible and efficient (Ye et al., 2016).

III. PRELIMINARIES

A. System Description

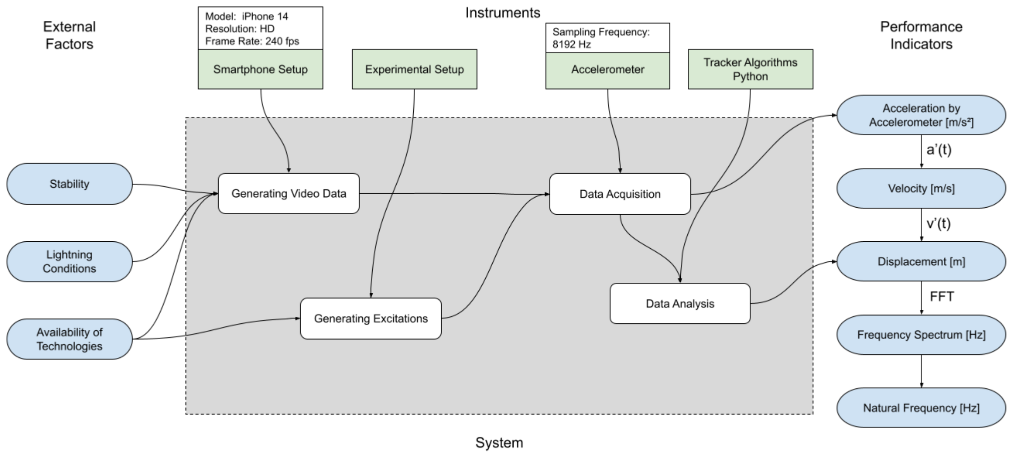


Fig. 1: Overview of the system. The system includes video data generation using a smartphone camera, excitation of the structural element, data acquisition from both the camera and an accelerometer, and data analysis using object-tracking algorithms to measure displacement and frequency.

The system representation focuses on using vision-based technologies to measure structural vibration and includes the following subsystems:

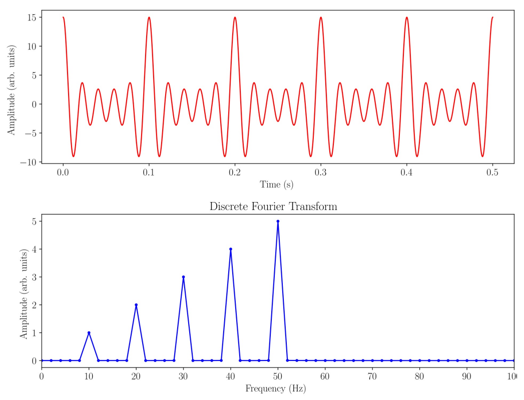
- Generating Video Data: This subsystem uses the integrated camera of a smartphone to record video data of the cantilever beam, during vibrations. The availability of advanced smartphone technologies supports it. Moreover, the lighting conditions will influence the quality of the recorded video.
- Generating Excitations: This subsystem is the practical setup where the structural element is exposed to an to generate a vibration for analysis.
- Data Acquisition: The raw vibration data is collected through the smartphone camera, which is then supported by an accelerometer to provide the reference measurement: acceleration. The generation of the video and excitations influence the quality of data collection.
- Data Analysis: Raw data is processed and analysed using object tracking algorithms. The analysis focuses on extracting the displacement over time by tracking a fixed point on the cantilever beam.

The key performance indicators (KPIs) in Figure 1 serve as the system's measurable outputs to evaluate the effectiveness of the vibration analysis of the different tracker algorithms. The KPIs provide quantifiable metrics, which are crucial for assessing the performance of the vibration measurement system. The displacement over time is determined by taking the derivatives of the acceleration over time measured by the accelerometer. As shown in figure 1.

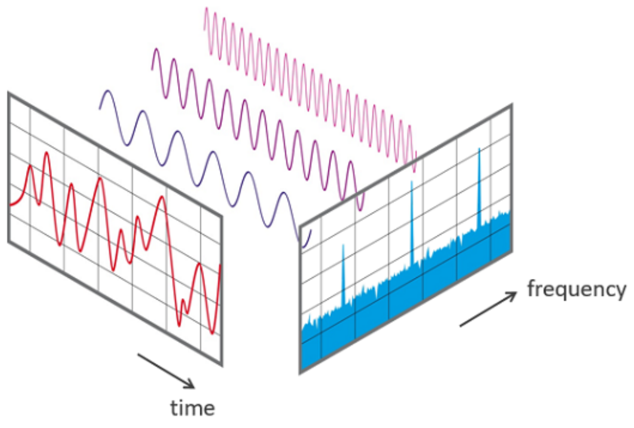
$$a''(t) = v'(t) = x(t) \tag{1}$$

The displacement over time is extracted from the video data using the object tracking algorithms. The fast Fourier transform (FFT) is used to determine the frequency domain and natural frequency.

$$\text{time domain } x(t) \xrightarrow{\text{FFT}} \text{frequency domain } x(f) \tag{2}$$



(a) Demonstrates the conversion of a vibration signal (displacement over time) to the frequency domain using FFT. The example highlights how structural vibrations can be analyzed to identify primary frequency components. (“Wikipedia”, 2024)



(b) Another example showing the conversion process of a vibration signal to the frequency domain, emphasizing the practical application of FFT in analyzing structural vibrations for health monitoring purposes. (NTi Audio, 2024)

Fig. 2: Examples of Fast Fourier Transform (FFT) Conversion.

Figures 2 shows an example of converting a vibration (displacement over time) to frequency by FFT.

B. Sampling Frequency and Its Influence on the Frequency Domain

Sampling frequency is a critical parameter in vibration measurement and analysis, influencing the accuracy and resolution of the frequency domain representation. In the context of SHM, understanding the implications of sampling frequency is essential for ensuring precise and reliable measurements. This subsection explores the role of sampling frequency, comparing the high-frequency capabilities of an accelerometer with the relatively lower sampling rate of a smartphone camera.
The sampling frequency, or sampling rate, is the number of samples per second taken from a continuous signal to create a discrete signal. It is measured in Hertz (Hz). In vibration analysis, the sampling frequency determines the highest frequency that can be accurately captured, known as the Nyquist frequency, which is half the sampling rate. Accurate vibration analysis requires a sampling frequency at least twice the signal’s highest frequency component (Gohil, 2019).

1) Accelerometer vs. Camera Sampling Frequencies

This study used two primary data acquisition tools: an accelerometer and a smartphone camera. The accelerometer employed has a sampling frequency of 8192 Hz, while the smartphone camera operates at 240 frames per second (fps), equivalent to a sampling frequency of 240 Hz.

- Accelerometer Sampling Frequency:
  - The accelerometer’s high sampling frequency (8192 Hz) allows it to capture high-frequency vibration components up to 4096 Hz (Nyquist frequency). This high resolution enables detailed frequency domain analysis, capturing fine and high-frequency vibrations that might be crucial for detecting structural abnormalities. (Gohil, 2019)
- Camera Sampling Frequency:
  - The smartphone camera’s sampling frequency of 240 Hz limits its ability to capture high-frequency components to 120 Hz (Nyquist frequency). Although significantly lower than the accelerometer’s, the camera’s sampling frequency is sufficient for many practical applications,

especially applications where the primary interest is lower-frequency vibrations typically associated with large-scale structural movements. (Sheet, 2023)

2) Influence on Frequency Domain Analysis

The difference in sampling frequencies between the accelerometer and the smartphone camera has direct implications on the frequency domain analysis:

- Resolution and Accuracy:
  - The higher sampling frequency of the accelerometer provides greater resolution and accuracy in the frequency domain. It captures a wider range of frequencies, offering a more comprehensive view of the vibration spectrum.
    The lower sampling frequency of the camera results in a coarser frequency domain representation, potentially missing higher frequency components. However, it remains effective for analyzing lower frequency vibrations, often the most critical for SHM (Gohil, 2019) (Dataloggers, 2023).

- Nyquist Frequency:
  - The accelerometer's high Nyquist frequency (4096 Hz) ensures that high-frequency vibrations are accurately represented without distortion (Dataloggers, 2023).
    The camera's Nyquist frequency (120 Hz) imposes a limitation, meaning any frequency components above 120 Hz will not be accurately captured and could be aliased, distorting the frequency domain representation (Gohil, 2019).

C. Stakeholder Analysis

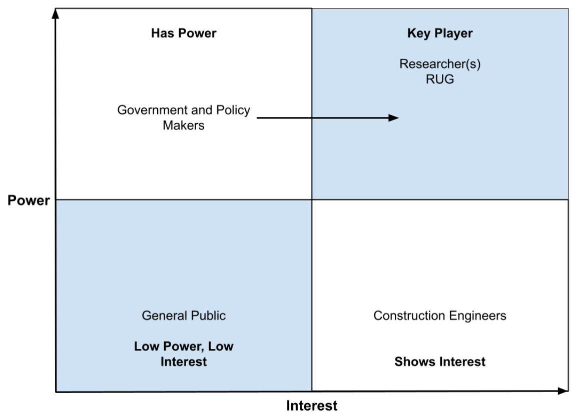


Fig. 3: Mendelow matrix including the different stakeholders in the research.

- *Government and Policy Makers:*
  The implementation of advanced SHM technologies like vision-based systems can significantly enhance structural safety, reduce maintenance costs, and ensure compliance with evolving safety standards. These outcomes are critical for public welfare and economic viability, aligning with national safety standards (Payawal & Kim, 2023). The government and policy makers, initially positioned in 'Has Power', may see their interest grow as the project progresses. As regulations develop, aligning the research with these changes could elevate the government's role from a powerful overseer to an active participant in promoting and mandating the use of vision-based SHM technologies (Abdulkarem et al., 2020).

- *Researcher(s) at the RUG:*
  Researchers are vital players with a substantial stake in the project's success, potential for publication, and further academic exploration. Their stakes lie in the project's ability to contribute novel insights into SHM and validate new methodologies.
- *Construction Engineers:*
  The research findings' practical applicability, reliability, and cost-effectiveness are essential for construction engineers. They are interested in solutions that can be integrated into existing workflows, enhance structural safety, and provide economic benefits through the efficiency of structures. Successful integration of vision-based SHM into existing workflows can enhance structural safety and provide economic benefits by reducing the need for manual inspections

D. Why-What Model

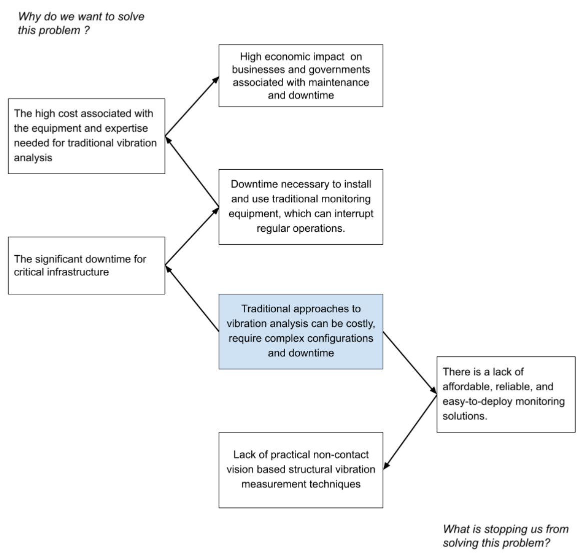

Why do we want to solve this problem ?
High economic impact on businesses and governments associated with maintenance and downtime
The high cost associated with the equipment and expertise needed for traditional vibration analysis
Downtime necessary to install and use traditional monitoring equipment, which can interrupt regular operations.
The significant downtime for critical infrastructure
Traditional approaches to vibration analysis can be costly, require complex configurations and downtime
There is a lack of affordable, reliable, and easy-to-deploy monitoring solutions.
Lack of practical non-contact vision based structural vibration measurement techniques
What is stopping us from solving this problem?

Fig. 4: Why-What model, showing the original problem and enhancing the understanding of this problem.

The Why-What Model is an analytical framework that helps understand the underlying problem and explains the steps taken to address it (Verschuren et al., 2010).

1) Why

The primary motivation behind employing the Why-What Model in this research stems from the limitations associated with traditional single-point vibration measurement methods. These traditional methods often necessitate direct contact with the monitored structure, leading to significant challenges such as high operational costs, complex setups, and unavoidable downtime during measurements. Additionally, data acquired from limited locations may provide an incomplete view of the structural vibrations, potentially overlooking broader issues affecting the entire structure.

2) What

To address these limitations, this research proposes using vision-based monitoring techniques, specifically by applying smartphone camera object-tracking algorithms. The Why-What Model guides the research process by identifying the core problem and the proposed solution. Here, the core problem is the inefficiency and impracticality of traditional single point-based vibration measurement methods. The proposed solution involves using advanced object-tracking algorithms to capture structural vibrations non-contact, thus providing a more efficient, cost-effective, and comprehensive approach to SHM.

IV. Problem Statement

Single-point-based methods often require direct contact with the structure and present substantial challenges, including high costs, complex setups, and operational downtime. These limitations hinder the practicality and accessibility of structural health monitoring. Moreover, these methods provide data from limited locations, restricting the analysis to specific points and potentially overlooking broader vibration issues across the entire structure.

V. Research Objective

This research aims to analyse and compare the accuracy of object-tracking algorithms with accelerometer measurements for vision-based vibration analysis on a cantilever beam, focusing on the accuracy of the displacement and frequency over time. The research planned to be completed within a 12-week period.

VI. Research Questions

A. Main Research Question

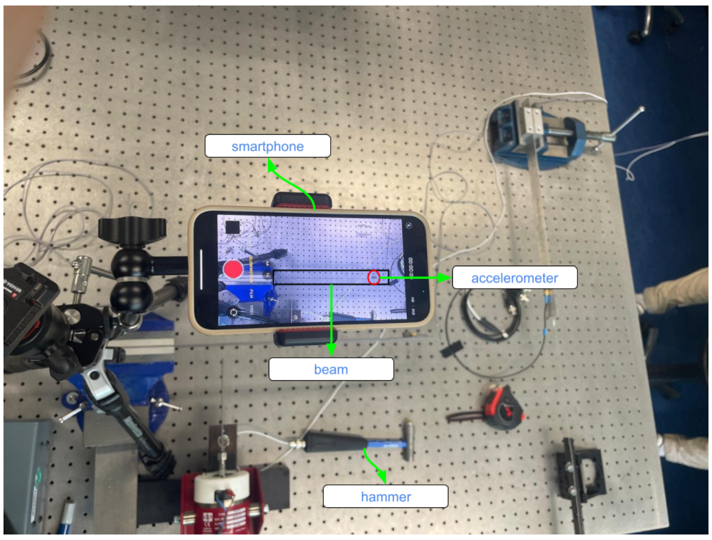
How does the accuracy of object-tracking algorithms compare with accelerometer measurements in both time and frequency domains on a cantilever beam, and which object-tracking algorithm proves most effective?

B. Sub-Questions

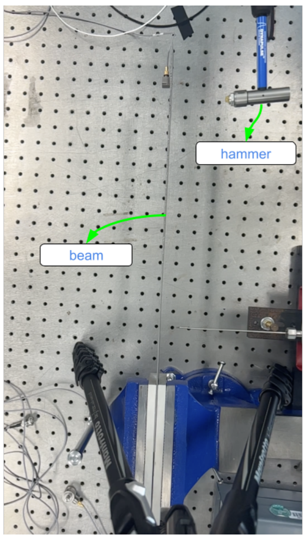
- *What object-tracking algorithms have been effectively utilized in similar studies for processing vision-based data?*
- *Which object-tracking algorithms are best suited for analyzing the data collected through smartphone-based vibration analysis on a cantilever beam?*
- *How does an impulse excitation affect the measurement accuracy of smartphone-based vibration analysis on a cantilever beam?*
- *Which of the tested object-tracking algorithms demonstrates the highest accuracy on impulse testing in smartphone-based vibration analysis on a cantilever beam compared to the accelerometer's reference measurements?*

VII. METHODS AND TOOLS

A. *Experimental Setup*



(a) Shows the overall experimental setup with a cantilevered beam subjected to vibrations, captured by a smartphone camera fixed on a stable tripod. (Side View)



(b) Shows the hammer setup used to generate impulse vibrations on the cantilever beam. (Top-Bottom View)

Fig. 5: Experimental Setup for Vision-Based Vibration Measurement.

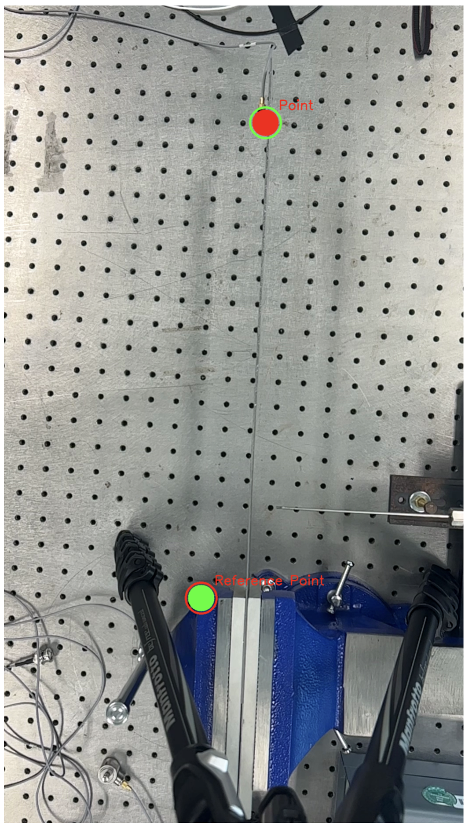
Central to the experiment is a setup with a cantilevered beam subjected to various generated vibrations captured with the smartphone camera fixed on a stable tripod, as shown in figure 5a. The hammer, as displayed in figure generates an impulse that exerts a vibration effect on the cantilever beam. The accelerometer's reference measurements and the video data are essential for assessing the frequency response accuracy.

1) Impulse Testing

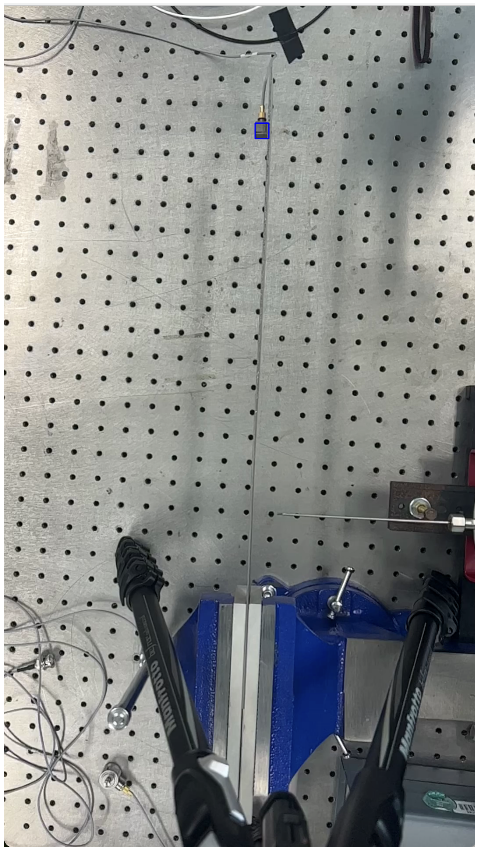
The unique characteristic of impulse testing is its ability to generate a complex frequency spectrum. An impulse force is a non-periodic pulse that simultaneously excites all the structure's natural frequencies. This broad frequency range is captured due to the short duration and high energy of the impulse, which induces vibrations across the entire spectrum of the structure's modal frequencies (Gohil, 2019) (Dataloggers, 2023).

B. *Data Analysis*

After acquiring the data, the focus shifts towards comparing object-tracking algorithms. In this phase, Open Source Computer Vision (OpenCV) object-tracking algorithms are integrated in Python. This integration includes edge detection and other tracker variations essential for extracting vibration data from recorded videos. The edge detection algorithms identify the boundaries of vibrating structures, and the trackers follow the movement throughout the experiment, as illustrated by Figure 6b. The tracker algorithms provided by OpenCV will allow visualization of the extracted displacement in both time and frequency domains, capturing the dynamics of structural vibrations of the cantilever beam. (Sharma et al., 2021)



(a) Object tracking using the Optical Flow (OF) algorithm. (Top-Bottom View)



(b) Object tracking using CSRT, MIL, and KCF algorithms. (Top-Bottom View)

Fig. 6: Application Visualization of Object Tracking Algorithms

Upon completion of the experiment and analysis, the accuracy of the graphs derived from the algorithm will be benchmarked against those from the accelerometer, with the most closely matching profile considered the most accurate.

Sub-Question	**Method**	**Tool**	**Milestone**
1	Evaluation of currently utilized object-tracking algorithms	Literature Review	Compilation of a list of object-tracking algorithms for vision-based data processing
2	Analysis of algorithms' suitability for smartphone-based analysis	Literature Review	Selection of optimal algorithms for smartphone-based vibration analysis on a cantilever beam
3	Impact assessment of different excitations on measurement accuracy	Empirical Testing using Python	Determination of excitation types of impact on the accuracy of vibration analysis
4	Comparative analysis of object-tracking algorithms' accuracy	Empirical Testing and Data Analysis with Python	Identification of the most accurate object-tracking algorithm for various excitations

TABLE I: Overview of Methods, Tools, and Milestones for Object-Tracking Algorithm Analysis.

1) Signal-to-Noise Ratio (SNR)

The SNR is essential for assessing how well object-tracking algorithms perform in vision-based structural vibration measurements. SNR measures the strength of the desired signal relative to the background noise, indicating the data's quality and reliability.

2) Calculation Method

The SNR is calculated as follows:

- Signal Power Calculation: The signal power is determined by averaging the squared values of the primary frequencies identified.
- Noise Power Calculation: The noise power is calculated by averaging the squared values of the frequency components that did not include the primary frequency peaks.
- SNR Formula: The SNR was calculated as the ratio of the signal power to the noise power, expressed in decibels (dB) using the formula:

$$\mathrm{SNR} = 10\log_{10}\left(\frac{\text{Signal Power}}{\text{Noise Power}}\right) \text{ (Xiu et al., 2023)}$$

How the SNR is incorporated in the MATLAB coding can be observed in the Appendix.

VIII. DELIVERABLE AND VALIDATION METHOD

The primary outcome of this research is a comparative analysis of different object-tracking algorithms using OpenCV and Python for structural vibration measurement via smartphone-based vision technology. This comparison not only evaluates these algorithms' capabilities and efficiencies but also validates the research methodology used. The accuracy and effectiveness of the algorithms, as measured against accelerometer measurements, validate the research's effectiveness. Hence, the research's validation is bound to the comparative accuracy of the object-tracking algorithms as benchmarked against known accelerometer measures.

IX. RESEARCH STRATEGY

The research strategy involves an assessment of different object-tracking algorithms to evaluate their efficiency in non-contact, vision-based structural vibration measurement. Each algorithm's advantages and drawbacks were analyzed to ensure a comprehensive comparison. Below, the methodology for each algorithm's application within the experimental setup is described.

A. Algorithm Description

In this research, an examination of various object-tracking algorithms is conducted to evaluate their efficiency in non-contact, vision-based structural vibration measurement. Each algorithm processes data differently, making them suitable for SHM applications.

1) CSRT (Channel and Spatial Reliability Tracking)

The CSRT tracker is known for its high precision and resilience in tracking rapidly moving objects, which are common in structural vibration scenarios. This tracker enhances traditional correlation filters by including spatial reliability, improving performance under various conditions. Its capability to accurately track objects even when they temporarily disappear from the view makes it particularly useful for SHM applications. (Muralidharan & Yanamadala, 2021)

2) MIL (Multiple Instance Learning)

MIL is included for its robustness in scenarios where the tracked object is partially obscured. MIL can maintain tracking accuracy even with intermittent visual obstructions by creating multiple potential positive models around the object's location. This capability is advantageous in practical SHM scenarios where environmental factors might partially block the structure being monitored. (Muralidharan & Yanamadala, 2021)

3) KCF (Kernelized Correlation Filter)

KCF is integrated into the strategy due to its speed and accuracy, essential for environments with rapid motion changes. Using machine learning classifiers trained on the target, KCF offers precise and efficient tracking. This makes it particularly suitable for SHM where quick and accurate detection of structural vibrations is critical. (Muralidharan & Yanamadala, 2021)

4) OF (Optical Flow)

The OF tracker calculates motion between frames based on object displacement, making it well-suited for analyzing continuous and smooth movements typical of structural vibrations. This method assesses motion by examining changes between consecutive frames, using a reference point to ensure accurate tracking. The reference point plays a crucial role. The reference point, marked in green in the setup in Figure 6a, maintains a consistent baseline against which the displacement of the measurement point, marked in red, is calculated. By tracking both the reference point and the measurement point across frames, the algorithm can determine the exact displacement and, thus, the vibration characteristics of the structure. (Xiu et al., 2023)

X. Results

A. *Acceleration obtained by Accelerometer during Impulse Test Analysis*

This section presents the acceleration collected by the accelerometer during an impulse test. This data is crucial as it serves as a benchmark for evaluating the performance of various object-tracking algorithms. The acceleration profile, illustrated in Figure 7, reflects the response of the cantilever beam to the impulse, providing a benchmark for comparisons.

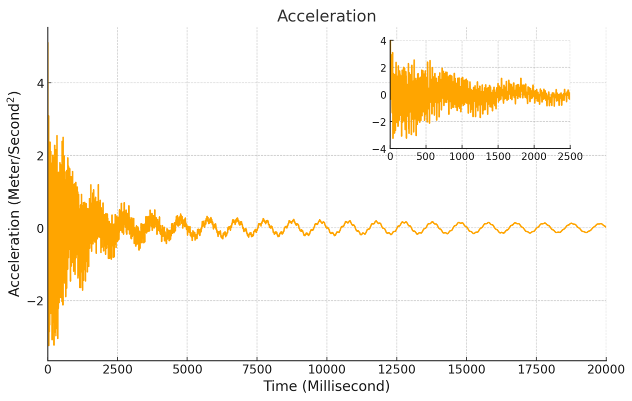
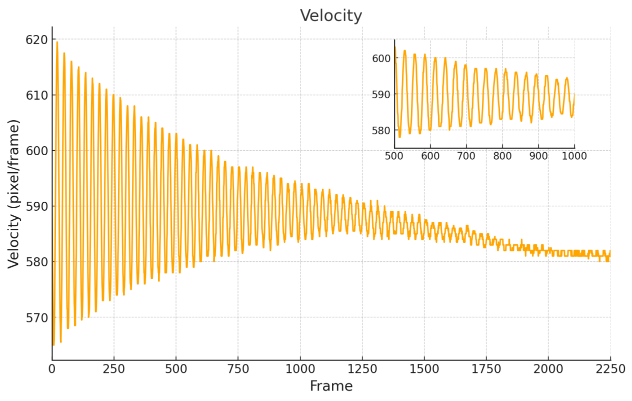


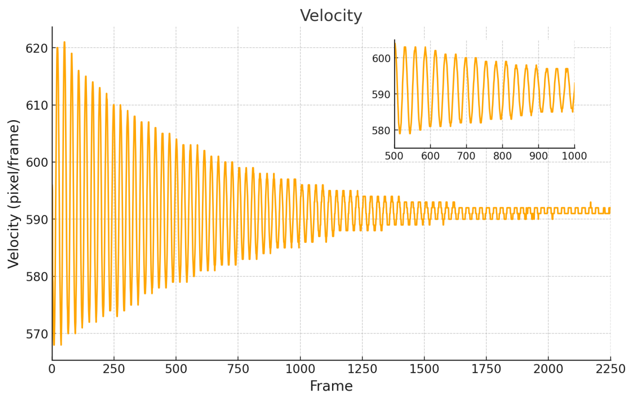
Fig. 7: This figure presents the acceleration data collected by the accelerometer during an impulse test on the cantilever beam. The profile reflects the beam's response to the impulse, serving as a benchmark for evaluating the performance of various object-tracking algorithms.

B. *Velocity obtained by Tracking Algorithms during Impulse Test Analysis*

This part examines the velocity profiles generated by different object-tracking algorithms during the impulse test. Figures 8 to 9 display the velocity profiles for each algorithm:

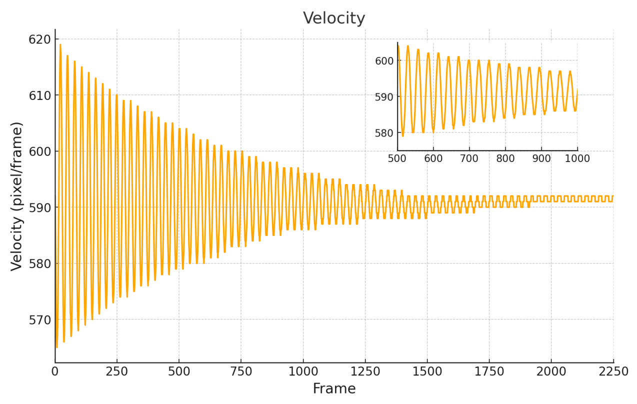


(a) Velocity profile obtained using the CSRT tracker during impact testing.

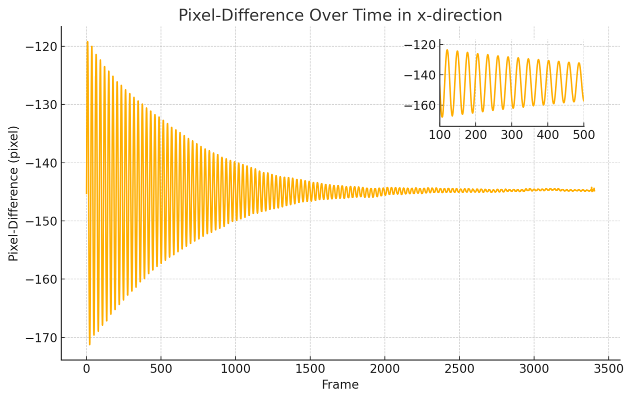


(b) Velocity profile obtained using the MIL tracker during impact testing.

Fig. 8: Velocity Profile csrt during Impact Testing; Velocity Profile MIL during Impact Testing.



(a) Velocity profile obtained using the KCF tracker during impact testing.



(b) Velocity profile obtained using the OF tracker during impact testing.

Fig. 9: Velocity Profile KCF during Impact Testing; Velocity Profile OF during Impact Testing.

The velocity plot for the CSRT algorithm shows a noticeable trend down towards the end of the measurement period. This decline indicates that the tracker lose accuracy or stability over time when following the structural vibrations. One factor could be the drift in the tracker's bounding box, leading to less accurate tracking as the sequence progresses. (Mallick, 2017) (SciTePress, 2020)

In the velocity plots for CSRT, MIL, and KCF, there is a particular "jumping" pattern. This behaviour is primarily due to the algorithms' dependence on integral pixel values for tracking. When the object's movement does not align with pixel boundaries, the algorithms can display sudden jumps or shifts in the tracked position. (Mallick, 2017) (SciTePress, 2020)

In contrast, the OF algorithm demonstrates smoother tracking in its velocity plot. The capability to handle sub-pixel movements effectively results in a more continuous and accurate representation of the object's velocity, free from the jumping patterns seen in the other algorithms (Xiu et al., 2023). The smooth tracking provided by the OF algorithm makes it particularly suitable for applications requiring precise measurement of continuous motion, such as in SHM.

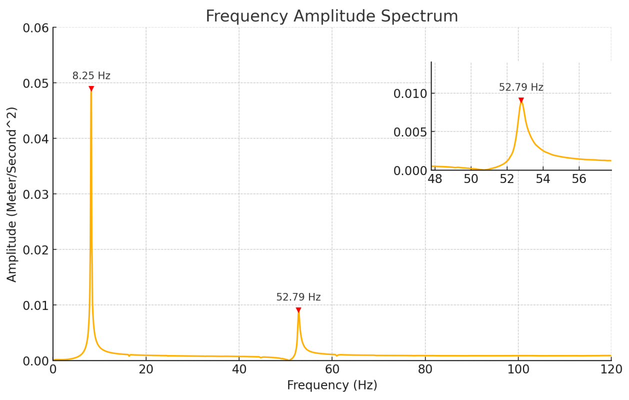
C. Frequency Domain Analysis

The upcoming subsections explain the frequency domain analysis of the data collected during the impulse test, using the FFT to convert time-domain signals into the frequency domain.

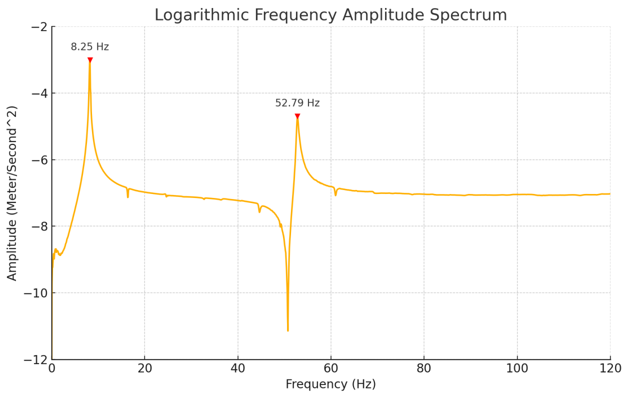
- Absolute Spectrum: Shows the magnitude of frequency components in the acceleration data, helping to identify the primary frequencies that describe the structural vibrations. The primary frequencies are obtained using the MATLAB MinPeakProminancy function. Additionally, the DC component, which represents the mean of the signal in the time domain, can be observed as the node around 0 Hz.
- Logarithmic Spectrum: Represents the frequency components on a logarithmic scale, making smaller magnitudes more visible. This is useful for detecting small frequencies that might not be seen in the absolute spectrum.

1) *Acceleration FFT*

Figures 10a and 10b display the FFT spectra of the acceleration data from the accelerometer:



(a) FFT of acceleration data collected by the accelerometer during impact testing. Identifies primary frequencies at 8.25 Hz and 52.79 Hz, serving as benchmarks for comparison.



(b) Logarithmic spectrum of the accelerometer's FFT data, enhancing visibility of smaller frequency components.

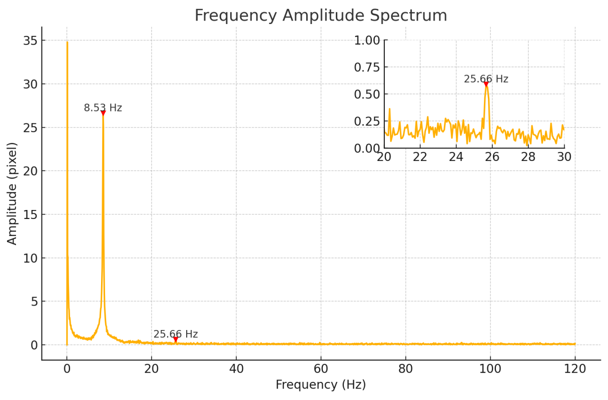
Fig. 10

- Peak 1: Frequency = 8.25 Hz
- Peak 2: Frequency = 52.79 Hz

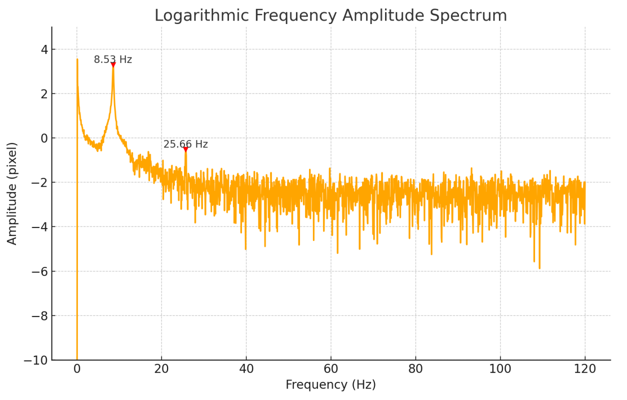
The peaks measured by the accelerometer will serve as a benchmark for the rest of the analysis.

2) *CSRT FFT*

Figures 11a and 11b show the FFT spectra of the velocity data obtained with the CSRT tracker:



(a) FFT of velocity data obtained with the CSRT tracker during impact testing. Identifies primary frequencies at 8.53 Hz and 25.66 Hz.



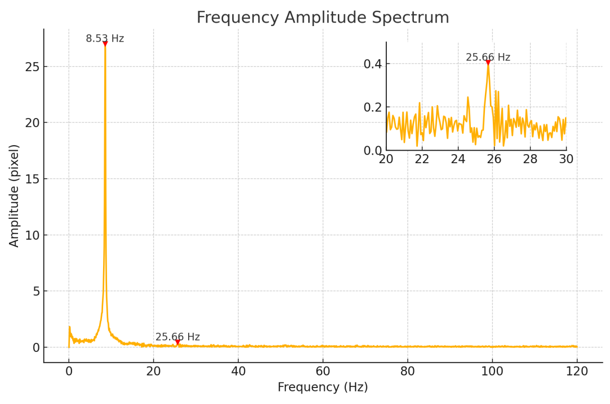
(b) Logarithmic spectrum of the CSRT tracker's FFT data, highlighting detailed frequency components.

Fig. 11

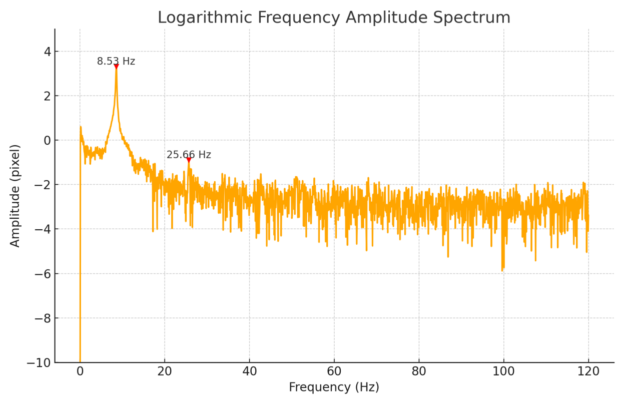
- Peak 1: Frequency = 8.53 Hz
- Peak 2: Frequency = 25.66 Hz

3) MIL FFT

Figures 12a and 12b present the FFT spectra for the velocity data obtained with the MIL tracker:



(a) FFT of velocity data obtained with the MIL tracker during impact testing. Identifies primary frequencies at 8.53 Hz and 25.66 Hz.



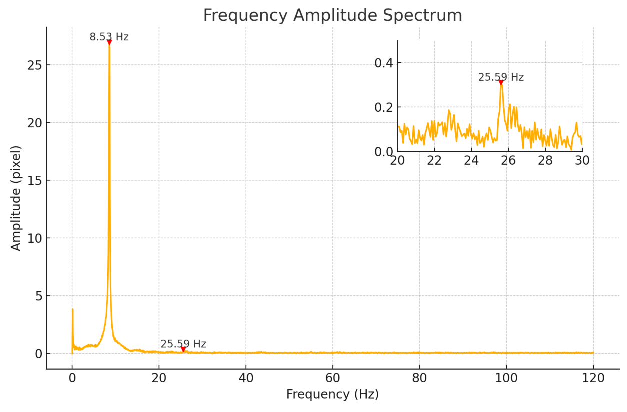
(b) Logarithmic spectrum of the MIL tracker's FFT data, highlighting detailed frequency components.

Fig. 12

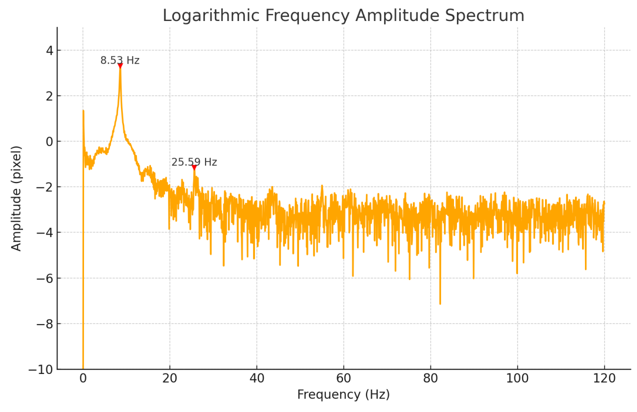
- Peak 1: Frequency = 8.53 Hz
- Peak 2: Frequency = 25.66 Hz

4) KCF FFT

Figures 13a and 13b display the FFT spectra for the velocity data obtained with the KCF tracker:



(a) FFT of velocity data obtained with the KCF tracker during impact testing. Identifies primary frequencies at 8.53 Hz and 25.59 Hz.



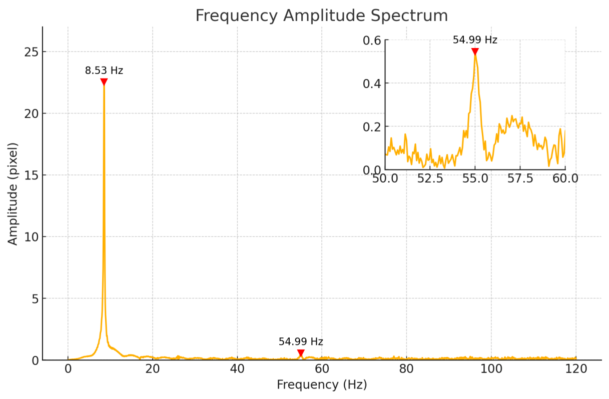
(b) Logarithmic spectrum of the KCF tracker's FFT data, highlighting detailed frequency components.

Fig. 13

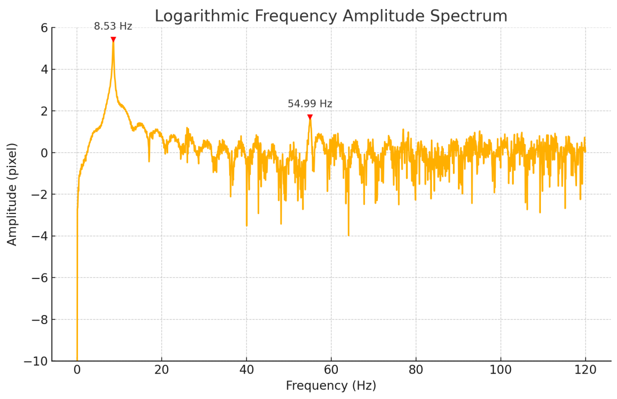
- Peak 1: Frequency = 8.53 Hz
- Peak 2: Frequency = 25.59 Hz

5) *OF FFT*

Figures 14a and 14b show the FFT spectra for the velocity data obtained with the OF tracker:



(a) FFT of velocity data obtained with the OF tracker during impact testing. Identifies primary frequencies at 8.53 Hz and 54.99 Hz.



(b) Logarithmic spectrum of the OF tracker's FFT data, highlighting detailed frequency components.

Fig. 14

- Peak 1: Frequency = 8.53 Hz
- Peak 2: Frequency = 54.99 Hz

D. SNR Results

CSRT	22.95 dB
MIL	23.88 dB
KCF	24.30 dB
OF	24.83 dB

TABLE II: Signal-to-Noise Ratio (SNR) of Different Object-Tracking Algorithms.

The SNR analysis in II provided insights into the performance differences among the various object-tracking algorithms. The SNR values indicate each algorithm's ability to distinguish the desired signal from background noise, thereby reflecting the data quality and reliability.

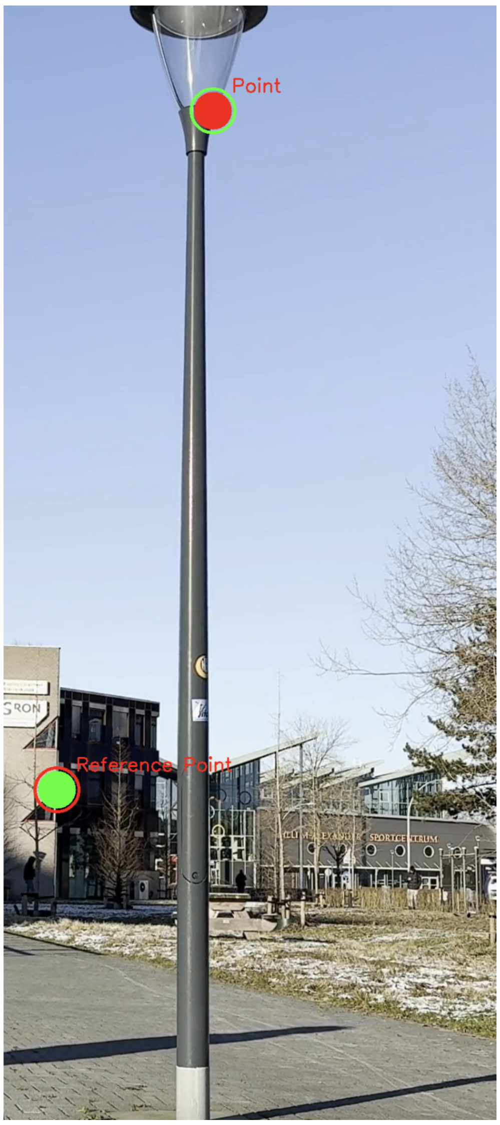
- CSRT Tracker: The CSRT tracker exhibited a moderate SNR of 22.95 dB, indicating a reasonable capacity for distinguishing the signal from noise. However, its effectiveness diminished in environments with higher noise levels compared to other trackers.
- MIL Tracker: The MIL tracker demonstrated an improved noise resilience with an SNR of 23.88 dB. Despite this, its precision in identifying smaller frequency components was limited.
- KCF Tracker: The KCF tracker achieved a high SNR of 24.30 dB, showcasing strong performance in environments characterized by rapid motion changes. It effectively minimized noise interference while accurately capturing the primary vibration signals. However, similar to the MIL tracker, it struggled to capture smaller frequency nodes effectively.
- OF Tracker: The OF tracker recorded the highest SNR of 24.83 dB, demonstrating its ability to track continuous movements and differentiate noise. Its high SNR and capability to accurately capture primary and secondary frequencies made it the most effective algorithm for vision-based structural vibration measurement among the algorithms tested.

XI. REAL-WORLD APPLICATION

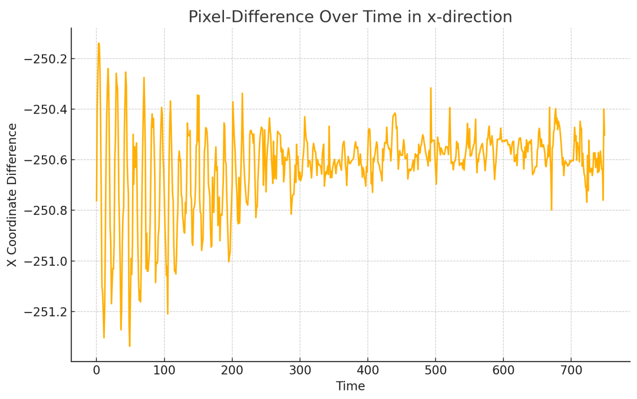
To further validate the OF tracker, a practical test at Zernike Campus was conducted on a lamppost subjected to impulse vibrations caused by a physical kick. This section outlines the setup, methods, and findings from this test.

A. Setup

A smartphone was secured on a stable tripod to film the lamppost, which was kicked to induce an impulse vibrations. The camera was positioned to maintain a clear and unobstructed view of the lamppost's motion.



(a) Experimental setup where a smartphone is secured on a stable tripod to film the lamppost. The figure highlights the point and the reference point used in the OF algorithm.

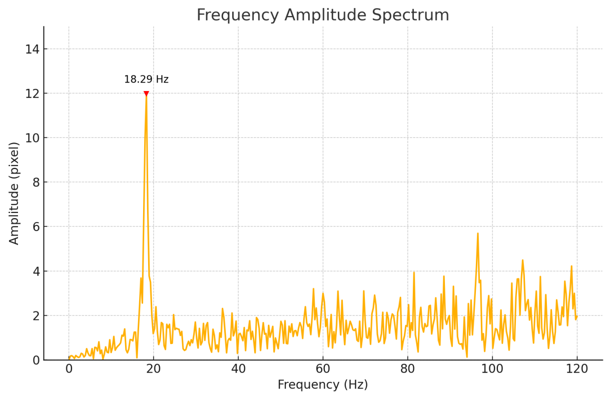


(b) Pixel difference over time between the reference point a the measured point on the lamppost.

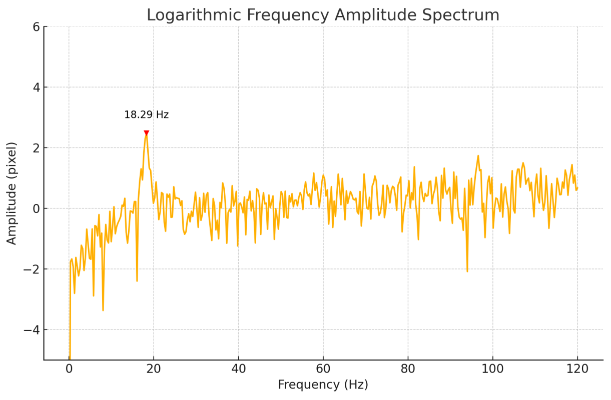
Fig. 15: Real-World Application on Lamppost Subjected to Impulse Vibrations

B. Velocity Analysis

The tracker effectively recorded the lamppost's velocity changes, providing a detailed view of how the velocity varied over time.



(a) Frequency amplitude spectrum of the lamppost's velocity data obtained during impact testing.



(b) Logarithmic frequency amplitude spectrum, enhancing the visibility of smaller frequency components in the lamppost's vibration data.

Fig. 16: Frequency Analysis of Lamppost Vibrations Using FFT

C. Frequency Analysis

The FFT analysis of the velocity data identified the primary frequency node of the lamppost's vibrations.

- Primary Frequency Node = 18.29 Hz

XII. DISCUSSION

A. *Algorithm Performance Analysis*

The comparative analysis of object-tracking algorithms highlights advancements in computer vision tools that enhance non-contact measurement techniques. The CSRT, MIL, KCF, and OF trackers demonstrated unique capabilities and limitations.

1) CSRT Tracker

- Exhibited moderate accuracy, with an SNR of 22.95 dB.
- Showed a noticeable decline in tracking accuracy over time due to drift in the tracker's bounding box.
- Effective in certain conditions but less reliable in maintaining stability throughout the measurement period.

2) MIL Tracker

- Demonstrated higher noise resilience with an SNR of 23.88 dB.
- Effective in scenarios with intermittent visual obstructions but lacked precision in identifying smaller frequency components.
- Robust against environmental factors but less accurate in fine-detail tracking.

3) KCF Tracker

- Achieved a high SNR of 24.30 dB, indicating strong performance in environments with rapid motion changes. Effective in minimizing noise interference while accurately capturing primary vibration signals. Similar to MIL, it failed to capture small frequency nodes effectively.

4) OF tracker

- Achieved the highest SNR of 24.83 dB, demonstrating exceptional ability in tracking continuous movements and differentiating noise.
- Closely matched the accelerometer's frequency components, making it the most accurate algorithm tested.
- Its ability to handle sub-pixel movements and smooth tracking makes it particularly suitable for SHM applications.

B. *Frequency Domain Analysis*

The FFT analysis provided a comprehensive view of the vibration frequencies captured by each tracking algorithm. Key findings include:

1) Accelerometer Benchmark Frequencies

- Primary Peak: 8.25 Hz
- Secondary Peak: 52.79 Hz

2) CSRT, MIL, and KCF Trackers

- Identified primary frequencies around 8.53 Hz, but deviated significantly in secondary frequencies, highlighting their limitations in capturing the complete vibration profile.

3) OF tracker

- Identified frequencies at 8.53 Hz and 54.99 Hz, closely matching the accelerometer benchmark and demonstrating superior accuracy in frequency domain analysis.

C. *Future Work*

Future research could apply these findings to various structural elements and environmental conditions. Additionally, further integrating machine learning techniques to refine the object-tracking algorithms could enhance their accuracy and adaptability. The continuous development of smartphone camera technology, particularly the increase in fps, promises to expand the capabilities and precision of vision-based SHM systems by increasing the sampling frequency and, therefore, the Nyquist frequency domain.

XIII. CONCLUSION

This research evaluated the efficiency of various object-tracking algorithms compared to traditional accelerometer measurements in non-contact, vision-based structural vibration analysis. Specifically, the study focused on validating the use of a smartphone for monitoring vibrations on a cantilever beam, assessing data quality in both time and frequency domains.

A. Algorithm Performance

The OF tracker demonstrated the highest accuracy and sensitivity among the tested algorithms. This was evident from its excellent SNR and precise identification of primary frequency components. The SNR results were as follows:

- CSRT: 22.95 dB
- MIL: 23.88 dB
- KCF: 24.30 dB
- OF: 24.83 dB

These results highlight the OF tracker's exceptional ability to capture signals and minimize noise interference, making it the most effective algorithm for vision-based structural vibration measurement.

B. Frequency Domain Analysis

FFT analysis provided a comprehensive view of the vibration frequencies captured by each tracking algorithm. The key frequencies identified by the accelerometer served as a benchmark:

- Primary Peak: 8.25 Hz
- Secondary Peak: 52.79 Hz

The FFT results for the tracking algorithms showed that the OF tracker closely matched the accelerometer's frequencies, with nodes at 8.53 Hz and 54.99 Hz. In contrast, algorithms CSRT, MIL, and KCF exhibited deviations, particularly in the smaller frequencies components:

- CSRT: 8.53 Hz, 25.66 Hz
- MIL: 8.53 Hz, 25.66 Hz
- KCF: 8.53 Hz, 25.59 Hz
- OF: 8.53 Hz, 54.99 Hz

These differences underscore the best frequency-capturing ability of the OF tracker, which successfully identified the primary frequency components with high accuracy.

C. Comparative Analysis

The velocity profiles and frequency domain analyses indicated that while all tracking algorithms could capture the essential dynamics of the vibrating structure, the OF algorithm consistently produced results closest to the accelerometer's reference measurements. This underscores its robustness in environments characterized by continuous, smooth motions typical of structural vibrations.

D. Practical Implications

The real-world application on a lamppost confirmed the OF tracker's ability to measure structural vibrations accurately in real-world scenarios. Its high sensitivity and precision in identifying key frequency node(s) make it an excellent tool for SHM, especially where traditional contact-based methods may be impractical or expensive.

References

Abdulkarem, M., Samsudin, K., Rokhani, F. Z., & A Rasid, M. F. (2020). Wireless sensor network for structural health monitoring: A contemporary review of technologies, challenges, and future direction. *Structural health monitoring*, *19*(3), 693–735.

Baqersad, J., & Di Maio, D. (2023). *Computer vision & laser vibrometry, volume 6: Proceedings of the 41st imac, a conference and exposition on structural dynamics 2023*. Springer Nature.

Dataloggers, C. (2023). Vibration measurement methods and techniques - cas dataloggers. https://dataloggerinc.com/

Feng, D., & Feng, M. Q. (2018). Computer vision for shm of civil infrastructure: From dynamic response measurement to damage detection–a review. *Engineering Structures*, *156*, 105–117.

Ferraris, C., Amprimo, G., & Pettiti, G. (2023). Computer vision and image processing in structural health monitoring: Overview of recent applications. *Signals*, *4*(3), 539–574.

Gohil, C. (2019, March). Vibration analysis basics – time waveform acquisition [in Predictive maintenance]. https://www.acoem.com/en/predictive-maintenance/vibration-analysis-basics-time-waveform-acquisition/

Mallick, S. (2017). Csrt — learnopencv [Accessed: 2024-06-12]. https://www.learnopencv.com/object-tracking-using-opencv-cpp-python/

Muralidharan, P. K., & Yanamadala, H. (2021). Comparative study of vision camera-based vibration analysis with the laser vibrometer method.

NTi Audio. (2024). Fast fourier transform (fft) [Accessed: 2024-06-08]. https://www.nti-audio.com/en/support/know-how/fast-fourier-transform-fft

Payawal, J. M. G., & Kim, D.-K. (2023). Image-based structural health monitoring: A systematic review. *Applied Sciences*, *13*(2), 968.

SciTePress. (2020). Object tracking using opencv [Accessed: 2024-06-12]. https://www.scitepress.org/Papers/2020/91838/91838.pdf

Sharma, A., Pathak, J., Prakash, M., & Singh, J. (2021). Object detection using opencv and python. *2021 3rd international conference on advances in computing, communication control and networking (ICAC3N)*, 501–505.

Sheet, E. C. (2023). Frequency measurement and sampling rate - engineering cheat sheet. https://engineeringcheatsheet.com/frequency-measurement-and-sampling-rate/

Tiboni, M., Remino, C., Bussola, R., & Amici, C. (2022). A review on vibration-based condition monitoring of rotating machinery. *Applied Sciences*, *12*(3), 972.

Verschuren, P., Doorewaard, H., & Mellion, M. (2010). *Designing a research project* (Vol. 2). Eleven International Publishing The Hague.

Wikipedia. (2024, March). https://en.wikipedia.org/wiki/Fast_Fourier_transform

Xiu, C., Weng, Y., & Shi, W. (2023). Vision and vibration data fusion-based structural dynamic displacement measurement with test validation. *Sensors*, *23*(9), 4547.

Ye, X.-W., Dong, C.-Z., Liu, T., et al. (2016). A review of machine vision-based structural health monitoring: Methodologies and applications. *Journal of Sensors*, *2016*.

Zhuang, Y., Chen, W., Jin, T., Chen, B., Zhang, H., & Zhang, W. (2022). A review of computer vision-based structural deformation monitoring in field environments. *Sensors*, *22*(10), 3789.

Appendix

A. Python Code for Trackers

```python
import cv2 as cv
import numpy as np
from scipy.io import savemat

cap = cv.VideoCapture(r"/Users/huibvanderveen/Desktop/Experiment/Experiment.MOV")
tracker = cv.TrackerKCF_create() #change different methods

ret, frame = cap.read()
bbox = cv.selectROI("Tracking", frame, False)
tracker.init(frame, bbox)
track_path = []
print("Successful")

while True:
    ret, frame = cap.read()

    if not ret:
        break

    ret, bbox = tracker.update(frame)

    if ret:
        (x, y, w, h) = [int(i) for i in bbox]
        track_path.append((x + w // 2, y + h // 2))

    if cv.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv.destroyAllWindows()

displacements = np.diff(track_path, axis=0)
matlab_data = {'csrt': displacements} #name of variation
savemat(r'/Users/huibvanderveen/Desktop/Experiment/KCF.mat', matlab_data)
#name of saving data
```

1) OF Tracker

```python
import cv2 as cv
import numpy as np
from scipy.io import savemat
import matplotlib.pyplot as plt

# Number of frames per second:
fpsVid = 30

cap = cv.VideoCapture("111.mp4")  # Replace with the correct path to your video file

frame_width = int(cap.get(3))
frame_height = int(cap.get(4))
size = (frame_width, frame_height)

_, frame = cap.read()

old_gray = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)

# Lucas Kanade parameters:
lk_params = dict(winSize=(20, 20),
                 maxLevel=5,
                 criteria=(cv.TERM_CRITERIA_EPS | cv.TERM_CRITERIA_COUNT, 10, 0.03))

```

```
points = []
output = []
fr = 0
pause = True

# Ask the user if a reference point is needed
print("Do you want to select a reference point? (y/n)")
user_input = input()
if user_input.lower() == 'y':
    reference_needed = True
else:
    reference_needed = False

# mouse callback function:
def select_point(event, x, y, flags, params):
    global points, point_selected, old_points, fr, pause
    if event == cv.EVENT_LBUTTONDOWN:
        points.append((x, y))
        print(f"Point {len(points)}: {points[-1]}")

        if reference_needed and len(points) == 2:
            point_selected = True
            old_points = np.array(points, dtype=np.float32)
            pause = not pause  # Toggle the pause state
        elif not reference_needed and len(points) == 1:
            point_selected = True
            old_points = np.array(points, dtype=np.float32)
            pause = not pause  # Toggle the pause state

cv.namedWindow("Frame")
cv.setMouseCallback("Frame", select_point)
point_selected = False
old_points = np.array([[]])

# Lists to store the x and y coordinates of the points for plotting
x_coords_point1 = []
y_coords_point1 = []
x_coords_point2 = []
y_coords_point2 = []

while True:
    try:
        if pause:
            cv.imshow("Frame", frame)
            key = cv.waitKey(30) & 0xFF
            if key == ord('q'):
                break
            continue

        _, frame = cap.read()

        if frame is None:
            print("Error: Failed to read frame.")
            break

        gray_frame = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)

        for i in range(len(old_points)):
            center = (int(old_points[i, 0]), int(old_points[i, 1]))
            color = (0, 255, 0) if i == 0 else (0, 0, 255)
            cv.circle(frame, center, 35, color, 3)

            # Display labels near the points
            label = "Reference Point" if reference_needed and i == 1 else "Point"
```

```python
            label_position = (int(old_points[i, 0]) + 30, int(old_points[i, 1]) - 30)
            cv.putText(frame, label, label_position, cv.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)

        new_points, status, error = cv.calcOpticalFlowPyrLK(old_gray, gray_frame, old_points, None, **lk_params)
        old_gray = gray_frame.copy()

        if status is not None and all(status):
            # Only update old_points and draw the circle if status is valid for all points
            old_points = new_points

            for i in range(len(new_points)):
                x, y = new_points[i].ravel()
                color = (0, 0, 255) if i == 0 else (0, 255, 0)
                label = 1 if reference_needed and i == 1 else 0
                cv.circle(frame, (int(x), int(y)), 30, color, -1)
                output.append([fr / fpsVid, x, y, label])

                # Append coordinates for plotting
                if i == 0:
                    x_coords_point1.append(x)
                    y_coords_point1.append(y)
                elif i == 1:
                    x_coords_point2.append(x)
                    y_coords_point2.append(y)

        fr += 1

        cv.imshow("Frame", frame)

        if cv.waitKey(10) & 0xFF == ord('q'):
            break
    except cv.error as e:
        print(f"Error: {e}")
        break

cap.release()
cv.destroyAllWindows()

# Plotting the points over time for two points
plt.figure(figsize=(15, 5))

# Plot x-coordinate over time for Point 1
plt.subplot(1, 3, 1)
plt.plot(range(len(x_coords_point1)), x_coords_point1, label='Point 1')
plt.title('X Coordinate Over Time for Point 1')
plt.xlabel('Frame Number')
plt.ylabel('X Coordinate')

# Plot y-coordinate over time for Point 1
plt.subplot(1, 3, 2)
plt.plot(range(len(y_coords_point1)), y_coords_point1, label='Point 1')
plt.title('Y Coordinate Over Time for Point 1 ')
plt.xlabel('Frame Number')
plt.ylabel('Y Coordinate')

# Plot the points themselves
plt.subplot(1, 3, 3)
plt.scatter(x_coords_point1, y_coords_point1, c=range(len(x_coords_point1)), cmap='viridis', marker='o', label='Point 1')
plt.title('Points Over Time')
plt.xlabel('X Coordinate')
plt.ylabel('Y Coordinate')
plt.colorbar(label='Frame Number')

plt.tight_layout()
plt.show()

# Plot Single-Sided FFT of point 1
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
fft_values_x = np.fft.fft(x_coords_point1)
freq_x = np.fft.fftfreq(len(fft_values_x), d=1 / fpsVid)
plt.plot(freq_x[:len(freq_x)//2], 20 * np.log10(np.abs(fft_values_x)[:len(fft_values_x)//2]))
plt.title('Single-sided FFT for Point 1 X Coordinate')
plt.xlabel('Frequency (Hz)')
plt.ylabel('Amplitude (dB)')

plt.subplot(1, 2, 2)
fft_values_y = np.fft.fft(y_coords_point1)
freq_y = np.fft.fftfreq(len(fft_values_y), d=1 / fpsVid)
plt.plot(freq_y[:len(freq_y)//2], 20 * np.log10(np.abs(fft_values_y)[:len(fft_values_y)//2]))
plt.title('Single-sided FFT for Point 1 Y Coordinate')
plt.xlabel('Frequency (Hz)')
plt.ylabel('Amplitude (dB)')

plt.tight_layout()
plt.show()



plt.figure(figsize=(15, 5))
# Plot x-coordinate over time for Point 1
plt.subplot(1, 3, 1)
plt.plot(range(len(x_coords_point2)), x_coords_point2, marker='o', label='Reference point')
plt.title('X Coordinate Over Time for Reference point')
plt.xlabel('Frame Number')
plt.ylabel('X Coordinate')

# Plot y-coordinate over time for Point 1
plt.subplot(1, 3, 2)
plt.plot(range(len(y_coords_point2)), y_coords_point2, marker='o', label='Reference point')
plt.title('Y Coordinate Over Time for Reference point ')
plt.xlabel('Frame Number')
plt.ylabel('Y Coordinate')

# Plot the points themselves
plt.subplot(1, 3, 3)
plt.scatter(x_coords_point2, y_coords_point2, c=range(len(x_coords_point1)), cmap='viridis',
    marker='o',
            label='Reference point')
plt.title('Points Over Time')
plt.xlabel('X Coordinate')
plt.ylabel('Y Coordinate')
plt.colorbar(label='Frame Number')
plt.tight_layout()
plt.show()

x_diff = np.array(x_coords_point2) - np.array(x_coords_point1)
y_diff = np.array(y_coords_point2) - np.array(y_coords_point1)
plt.figure(figsize=(20, 20))
plt.subplot(2, 2, 1)
plt.plot(range(len(x_diff)), x_diff)
plt.title('Pixel-Difference Over Time in x-direction')
plt.xlabel('X Coordinate')
plt.ylabel('Y Coordinate')

plt.subplot(2, 2, 2)
plt.plot(range(len(y_diff)), y_diff)
plt.title('Pixel-Difference Over Time in y-direction')
plt.xlabel('X Coordinate')
plt.ylabel('Y Coordinate')

# Plot Single-Sided FFT of point difference
plt.subplot(2, 2, 3)
fft_values = np.fft.fft(x_diff)
freq = np.fft.fftfreq(len(fft_values), d=1 / fpsVid)
plt.plot(freq[:len(freq) // 2], np.log(np.abs(fft_values)[:len(fft_values) // 2]), marker='o'
    )
plt.title('Single-sided FFT for relative motion X Coordinate')
plt.xlabel('Frequency (Hz)')
plt.ylabel('Amplitude')

plt.subplot(2, 2, 4)
fft_values = np.fft.fft(y_diff)
freq = np.fft.fftfreq(len(fft_values), d=1 / fpsVid)
plt.plot(freq[:len(freq) // 2], np.log(np.abs(fft_values)[:len(fft_values) // 2]))
plt.title('Single-sided FFT for relative motion Y Coordinate')
plt.xlabel('Frequency (Hz)')
plt.ylabel('Amplitude')
plt.show()


matlab_data = {'output': np.array(output)}
savemat('output_data.mat', matlab_data)
```

B. MATLAB Code for Evaluating and Plotting the CSRT, MIL and KCF Tracking Data

```
% Load velocity data from the first column
load('CSRT.mat');
velocity_data = CSRT(1,:);

% Plot velocity data highlighting the impact area
figure;
plot(velocity_data);  % Original velocity data in blue
xlabel('Frame');
ylabel('Velocity (pixel/frame)');
title('Velocity');

% Set x-axis limits to focus on the region around the impact
xlim([0 , 2250]);  % Adjust these values as needed

% Frequency analysis of the derived acceleration data
k = 240;
N = length(velocity_data);
centered_velocity = velocity_data - mean(velocity_data); % Centering the data
fft_velocity = abs(fft(centered_velocity))/N*2; % FFT of centered data
half_fft_velocity = fft_velocity(1:floor(N/2));
frequency_amplitude = half_fft_velocity .* 2 .* pi;

% Generate the tt vector to match the length of frequency_amplitude
tt = linspace(0, k/2, length(frequency_amplitude));

% Ensure no zero values before logarithm
log_frequency_amplitude = log(frequency_amplitude + 1e-6);  % Adding a small constant to
    avoid log(0)

% Make sure frequency_amplitude is a vector
frequency_amplitude = frequency_amplitude(:);  % Convert to column vector if necessary

% Finding and marking peaks - Regular Spectrum
[peak_values, peak_locations] = findpeaks(frequency_amplitude, 'MinPeakProminence', 6); %
    Tune these parameters

% Calculate SNR
signal_power = mean(peak_values.^2);
noise_power = mean((frequency_amplitude(setdiff(1:end, peak_locations))).^2);
SNR = 10 * log10(signal_power / noise_power);
disp(['SNR: ', num2str(SNR), ' dB']);

% Plotting frequency amplitude spectrum
figure;
plot(tt, frequency_amplitude);
xlabel('Frequency (Hz)');
ylabel('Amplitude (pixel)');
title('Frequency Amplitude Spectrum');
% Set y-axis limits to focus on the region around the impact
%ylim([0 , 300]);
%xlim([0 , 120]);

% Marking peaks
hold on;
plot(tt(peak_locations), peak_values, 'rv', 'MarkerFaceColor', 'r'); % Mark peaks with red
    inverted triangles
hold off;

% Annotating the peaks with their frequency values
for i = 1:length(peak_values)
    text(tt(peak_locations(i)), peak_values(i), sprintf('%.2f Hz', tt(peak_locations(i))), '
    VerticalAlignment', 'bottom', 'HorizontalAlignment', 'center');
```

```
end

% Plotting logarithmic frequency amplitude spectrum
figure;
plot(tt, log_frequency_amplitude);
xlabel('Frequency (Hz)');
ylabel('Amplitude (pixel)');
title('Logarithmic Frequency Amplitude Spectrum');
% Set y-axis limits to focus on the region around the impact
ylim([-10 , 5]);

% Finding and marking peaks in logarithmic data
log_frequency_amplitude = log_frequency_amplitude(:);  % Convert to column vector if
    necessary
[log_peak_values, log_peak_locations] = findpeaks(log_frequency_amplitude, 'MinPeakProminence
    ', 4.6); % Adjust 'MinPeakProminence' as needed
hold on;
plot(tt(log_peak_locations), log_peak_values, 'rv', 'MarkerFaceColor', 'r'); % Mark peaks
    with red inverted triangles
hold off;

% Annotating the peaks with their frequency values
for i = 1:length(log_peak_values)
    text(tt(log_peak_locations(i)), log_peak_values(i), sprintf('%.2f Hz', tt(
    log_peak_locations(i))), 'VerticalAlignment', 'bottom', 'HorizontalAlignment', 'center');
end
```

C. *MATLAB Code for Evaluating and Plotting the OF tracker Data*

```
% Load velocity data from the first column
x_coords_point1 = OFselected(:,1);
x_coords_point2 = OFreference(:,1);

y_coords_point1 = OFselected(:,2);
y_coords_point2 = OFreference(:,2);

x_diff = x_coords_point2 - x_coords_point1;
y_diff = y_coords_point2 - y_coords_point1;

% Plot velocity data highlighting the impact area
figure;
plot(x_diff);  % Original velocity data in blue
xlabel('Time');
ylabel('X Coordinate Difference');
title('Pixel-Difference Over Time in x-direction');

% Frequency analysis of the derived acceleration data
k = 240;
N = length(x_diff);
tt = linspace(0, k/2, floor(N/2)); % Frequency vector

% Ensure tt is a column vector
tt = tt(:);

centered_velocity = x_diff - mean(x_diff); % Centering the data
fft_velocity = abs(fft(centered_velocity))/N*2; % FFT of centered data
half_fft_velocity = fft_velocity(1:floor(N/2));
frequency_amplitude = half_fft_velocity .* (2 * pi * (0:floor(N/2)-1) / N * k);

% Ensure frequency_amplitude is a vector
frequency_amplitude = frequency_amplitude(:); % Ensure it is a column vector

% Ensure no zero values before logarithm
```

```
log_frequency_amplitude = log(frequency_amplitude + 1e-6);  % Adding a small constant to
    avoid log(0)

% Plotting frequency amplitude spectrum
figure;
plot(tt, frequency_amplitude);
xlabel('Frequency (Hz)');
ylabel('Amplitude (pixel)');
title('Frequency Amplitude Spectrum');

% Find the highest peak in the frequency amplitude spectrum
[~, peak_index] = max(frequency_amplitude);
peak_frequency = tt(peak_index);
peak_amplitude = frequency_amplitude(peak_index);

% Calculate SNR
signal_power = mean(peak_index.^2);
noise_power = mean((frequency_amplitude(setdiff(1:end, peak_index))).^2);
SNR = 10 * log10(signal_power / noise_power);
disp(['SNR: ', num2str(SNR), ' dB']);


% Plot the upside-down red triangle at the peak
hold on;
plot(peak_frequency, peak_amplitude, 'v', 'Color', 'red');  % 'v' for upside-down triangle

% Annotate the peak directly above the red triangle
text(peak_frequency, peak_amplitude + 0.5, sprintf('%.2f Hz', peak_frequency), 'Color', '
    black', 'HorizontalAlignment', 'center');
hold off;

% Plotting logarithmic frequency amplitude spectrum
figure;
plot(tt, log_frequency_amplitude);
xlabel('Frequency (Hz)');
ylabel('Amplitude (pixel)');
title('Logarithmic Frequency Amplitude Spectrum');

% Set y-axis limits to focus on the region around the impact
ylim([-5 , 10]);

% Ensure log_frequency_amplitude is a vector
log_frequency_amplitude = log_frequency_amplitude(:); % Ensure it is a column vector

% Plot the upside-down red triangle at the peak in the logarithmic plot
hold on;
plot(peak_frequency, log_frequency_amplitude(peak_index), 'v', 'Color', 'red');  % 'v' for
    upside-down triangle

% Annotate the peak directly above the red triangle in the logarithmic plot
text(peak_frequency, log_frequency_amplitude(peak_index) + 0.5, sprintf('%.2f Hz',
    peak_frequency), 'Color', 'black', 'HorizontalAlignment', 'center');
hold off;
```