Video Motion Magnification

Abstract

In this hackathon project, we present a comprehensive software suite for Video Motion Magnification (VMM) with a focus on practical implementation. VMM is a valuable technique for detecting subtle vibrations and motions in various applications, from engineering to healthcare. Our software suite incorporates algorithms for VMM, time-series data extraction from videos, and Fast Fourier Transform (FFT) processing of time-series data. The suite features a user-friendly graphical user interface (GUI), making it accessible to users with no prior signal processing experience.

The key contributions of this project include the implementation of the efficient Eulerian approach for VMM and the development of algorithms that enable users to magnify and analyze motion in videos. By selecting appropriate frequency bands and parameters, users can reveal crucial information about the systems they are studying.

The practical significance of our project lies in the cost-effectiveness and efficiency of VMM compared to traditional vibration analysis methods. By utilizing video imaging, we eliminate the need for specialized hardware, making VMM accessible for a wide range of applications. The GUI enhances usability, allowing users to visualize, edit, and export data easily.

While challenges and limitations, such as hardware constraints and noise in real-world scenarios, exist, this project represents a promising step forward in the field of VMA. Future directions include hardware improvements, noise reduction techniques, and integration with IoT devices. Overall, our software suite offers a versatile and powerful tool for real-world vibrational analysis needs.

1. Introduction

Shuttle motion yet insignificant in human perception plays very important role in the world. The vibration of a revving engine and the shaking of a bridge, they all carry vital information about the system they are part of. There are various applications of this vibration analysis in different fields starting from mechanical systems to structures and even in medical domain. Vibration can help us understand the structural integrity in civil engineering.

Currently, various in-situ devices are used to get the reading of the vibration of the system of interest. For examples accelerometers are used which are placed on the surface and they detect the vibrations. But it's not always possible to place accelerometers at any point of interest. Various strain measuring sensors are used in the field of civil engineering which track the minute displacements in the system.

In the recent years, with the advent of higher computing power and rise of computer vision newer techniques have been developed to use video imaging to find out the vibrations in objects and surfaces. The consecutive frames hold the information of movement as the pixels shift in with motion. This shuttle information can be used to analyses the motion. However, these techniques are not as accurate as the above-mentioned traditional ones. Yet they have come a long way. They provide a very good analysis within very less time and cost. Video Motion Magnification (VMM) only require a video of the object and no consumables are used. This makes it very cost effective and quick, compared traditional techniques which take more time to assemble and require consumables.

2. Objective

Our objective in this project is to create a full-fledged software which is equipped with all the necessary tool to perform vibrational analysis of the given video. This suite includes VMM, frequency domain analysis, optical-flow calculation, segregation of motion based on

difference in frequency and motion pixel identification with grayscale and color frames. We have utilized various novel techniques to achieve the above. These algorithms are computationally very heavy, hence for a practical and functioning system we also aim to optimize the process-flow and aim use multi-threading and multiprocessing to use best of the machine the software runs on. We also use media pipeline to prevent loading excessive data to the RAM and provide streamlined performance.

All these experiences demand a graphical-user-interface (GUI) such that anyone even with no experience in processing running under the hood can utilize the finicalities given to the fullest. The GUI is very easy to grasp and yet effective simultaneously.

3. Methodology

As mentioned above there are two main approaches in the VMM, Lagrangian and Eulerian. We have selected Eulerian approach for this project. This approach consumes less computation and is pretty straight forward. It was published in a MIT paper titled *"Eulerian Video Magnification for Revealing Subtle Changes in the World"* in SIGGRAPH 2012. This approach is based on amplifying the temporal changes in the video and creating a resultant video with amplified motion. Details of the approach is mentioned in later sections.

In our application, user has to select a video file. Then he/she may specify the parameters and start the magnification of video. From the generated video user can pick any point in the frame and then a time series is created for that particular region which contains the change in the pixel values in the same. This time series corresponds to the motion in that location of the frame. The time-series is then processed through FFT, the generated frequency domain signal presents the dominant frequencies in the signal. Apart from the above VMM, there are other options like generating difference video of successive frames. The application also provides feature to edit

and plot generated data. One can also export the generated result in their format of choice like csv, xlsx, etc.

There are three main algorithms that form the backbone of the software, they are for VMM, generating time-series data from video, creating FFT of the time-series. All of this is encapsulated in a GUI that makes it accessible and easy to use.

3.1. VMM

For VMM, as we have mentioned earlier we are using Eulerian Motion Amplification for amplification if the motion. This technique was published in the MIT paper. It takes in standard video. The magnification is performed in a specific band of frequency. Based on the step difference provided and the starting and end frequency, one or multiple bands are created with equal difference. Then the algorithm is run through all the bands and the magnified video is stored.

For magnification, this algorithm performs spatial decomposition of the frames. These frames are successively down-sampled in Laplacian pyramid. Temporal processing is performed on the resultant. For an image, say in one dimension, $I(x,t)$ represents the intensity, with position *x* and time *t*. Then, intensity can be represented as shown in Eq.1.

$$I(x,t) = f(x,\delta(t)) \tag{1}$$

Since, we want to amplify the motion part of the intensity so we need to generate a new $\bar{\mathrm{I}}(x,t)$, as in Eq.2.

$$\bar{\mathrm{I}}(x,t) = f(x,(1+\alpha)\delta(t)) \tag{2}$$

where α is the amplification factor.

Since, we have several temporal sub-bands, the total amplified video frame can be given Eq.3.,

$$\bar{\mathrm{I}}(x,t) = f(x + \sum_{k}(1+\alpha_k)\delta_k(t)) \tag{3}$$

3.2. Frequency Analysis

Each moving frame record a specific state of the object. If we pick a small region of a grayscale frame, the sum of all the pixels in the region represent the intensity in the region. Due to motion, assuming vibrational/periodic, the intensity of the region also oscillate with the same frequency. So, if we pick a point (x, y) and length (m, n), then the intensity can be written as shown in Eq.4,

$$J(x,y) = \sum_{i=x}^{x+m}\sum_{j=y}^{y+m} I_{ij} \tag{4}$$

where I_{ij} represent the intensity of the (i, j) th pixel of the grayscale image I.

This technique is very straight forward. One drawback of this method is that it's very difficult to get accurate motion/ time-series data from homogenous surfaces. The change in such regions become very small and thus it can sometime lead to inaccurate result in the time-series.

However, this technique has worked fine with judicious selection of the region for obtaining time-series data. Also creating contrast with attaching something of different color can also mitigate the problem.

3.3. Optical Flow

The optical flow represents the direction of motion in successive frames. This assumes that overall brightness or intensity of the image is constant. So, for any 2D and time dependent image sequence we may say, Eq.5,

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t) \tag{5}$$

Taking the linear approximation of above equation leads to Eq.6 and Eq.7,

$$\frac{\Delta x}{\Delta t}\frac{\partial I(x, y, t)}{\partial x} + \frac{\Delta y}{\Delta t}\frac{\partial I(x, y, t)}{\partial y} + \frac{\partial I(x, y, t)}{\partial t} = 0 \tag{6}$$

$$\Rightarrow u(x, y, t)\frac{\partial I(x, y, t)}{\partial x} + v(x, y, t)\frac{\partial I(x, y, t)}{\partial y} + \frac{\partial I(x, y, t)}{\partial t} = 0 \tag{7}$$

Using Lucas–Kanade method, which assumes that in neighborhood of the point the optical flow *u(x, y, t)* and *v(x, y, t)*is constant. Thus, Eq.8. can be,

$$\begin{aligned} I_x(q_1)V_x + I_y(q_1)V_y &= -I_t(q_1) \\ &\vdots \\ I_x(q_n)V_x + I_y(q_n)V_y &= -I_t(q_n) \end{aligned} \tag{8}$$

where $q_1, q_2, \cdots, q_n$ are the pixels inside the window, and $I_x(q_i), I_y(q_i), I_t(q_i)$ are partial derivatives of the image *I* with respect to position *x, y* and time *t*. This system of equation is usually over-constrained, hence LK method used least square technique to solve this for approximating the value of V_x and V_y.

This vector is then plotted on to the frame as shown below. A filter is used to rule out zero or other small motions. This filter is controlled by the user as per the requirement. A color scheme can be applied to the vectors dawn it can also be changed in the setting. The length of the vector is given by Eq.9,

$$l = \Omega\sqrt{V_x^2 + V_y^2} \tag{9}$$

where Ω is the scale factor.

4. Validation Strategy

With the above methods in place we needed a way to validate our results. We used real-life videos for qualitative analysis and the computer-generated video for quantitative analysis. We recorded some of the videos ourselves with mobile-phone and some are standard videos that we downloaded from the internet. For computer generated video we used Blender, a popular open-source software.

4.1. Physical Video

We recorded some real-life videos involving some shuttle motion. Although we were not sure of the quantitative results, but the qualitative results came out to be pretty satisfying. We are working on a physical electromagnet based oscillator model where we can control the amplitude and the frequency of the motion. This model will be used to validate the results quantitatively. For the mean-time, we magnified the existing video and we also took some of the standard video from the internet for magnification.

4.2. Computer Generated Video

In Blender, we used a cube to simulate the motion. Any motion in the 3D space can be given by the Eq. 10,

$$s(t) = \left(a_x \sin(2\pi f_x t), a_y \sin(2\pi f_y t), a_z \sin(2\pi f_z t)\right) \tag{10}$$

where a_x, a_y and a_z represent the amplitude of the motion in the x, y and z direction respectively. f_x, f_y and f_z represent the frequency of motion in the respective directions.

To get the appropriate keyframe points to represent the above motion closely. We get *S(t)* given by Eq. 11,

$$S(t) = \sum s_i(t) \tag{11}$$

Now, the derivative of *S(t)* will be zero when the direction of motion changes, thus we get all the primary solutions of the Eq. 12,

$$\frac{dS(t)}{dt} = 0 \tag{12}$$

Since, *S(t)* is a sinusoidal function, it's derivative will also be sinusoidal. Due to this there will be infinitely many solutions of Eq. 12. But all the solutions less than the time-period of *S(t)*, mentioned as primary solutions, are used as keyframe points. We calculated all the *x, y* and *z* positions on these points and used them to keyframe the animation. Blender interpolates between these keyframes and generates a smooth motion. Further, noise is added to the motion through additional modifier. We created an infinite cycle with the Graph Editor (in Blender) and recorded the animation in desired frame-rate, aspect-ratio and image quality.

5. Implementation

The software is created to be modular and dynamic. We focused on using the efficient tools available or develop them ourselves. We used python libraries for the project. Although for developing a full-fledged application it's not the first choice. Various other alternatives like Java with cross-platform support and C++ with high control over level memory management could also be used for the same which would have had a more efficient result. But due lack of time we choose Python. All the necessary libraries are already available in Python for mathematical computation and computer vision. This enabled us to quickly prototype and develop a working model. And this decision turned out to be good as we were able to get it working very quick.

5.1. Computational Flow

In Python, two libraries form the backbone of all the computations, namely Numpy and OpenCV. Both of them are well established library with huge user base. All the secondary functions used in the software are based on these two libraries. The GUI is made with Tkinter library. Although there are other GUI libraries like ImGUI are available but we choose Tkinter as it saved time.

The GUI and the processing the data runs on separate thread allowing the software to poll events and respond accordingly. The GUI features a built in excel-like table renderer, based OpenPyXL. It helps to view and edit data directly inside the application with ease. It is also equipped with Matplotlib based graph plotter. It has most of the functionality of the Matlab's plotting tool. Hence, it provides a one stop solution for all of the analysis.

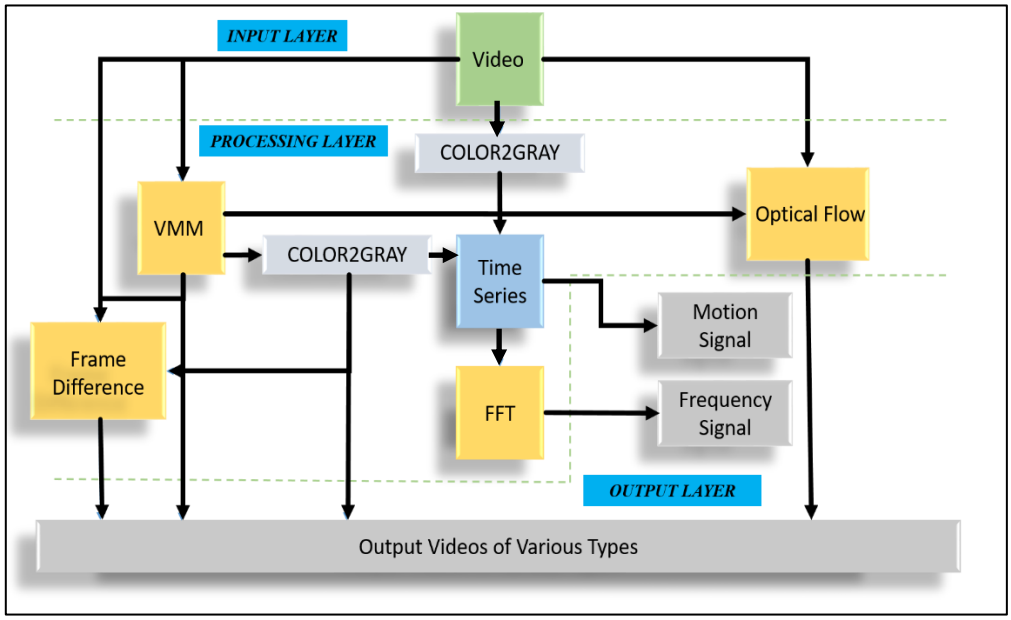

Fig.1 Computational Flow of the software

5.2. I/O model

Each project has a configuration file with ***.hwk*** extension. This file contains path to all the required files and directories. It also has the user entered parameters. So, for each new project a new project file is generated. One can also write their own project file with all the necessary rules. It also generates intermediate result files for the user. You can export different video files, as shown in above flowchart. These exports can be in various Video-Encodings like H.264, H.265 (HEVC), VP9, and AV1. The numerical results like that of power-spectral density, motion signal, etc. can be exported in various popular formats like csv, excel, etc.

6. Results and Discussion

Based on the above discussed approach we developed a barebone prototype for the software. It consisted a simple GUI for easy and quick use. We generated the results for analysis and validation of the algorithms envisioned for use in the process. We focused on both the qualitative and quantitative analysis of the output. As discussed earlier, we used the amplified video of the physical recorded or downloaded video for qualitative analysis and computer-generated video for the qualitative analysis. Through computer generated video, we could control the motion and the noise in it precisely for validating the generated output.

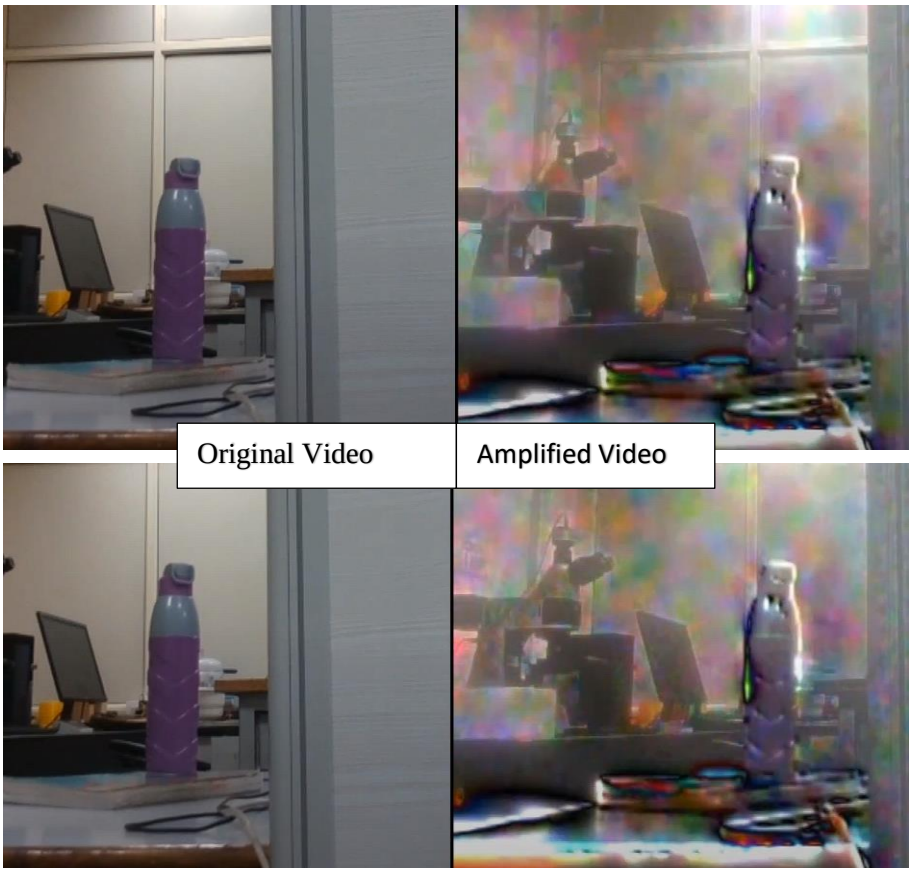

Fig.2 Comparison between original and amplified video

The amplified video shows significant amplification of the motion in the specified frequency band. Fig.2 compares the two videos side by side. The amplified video has increased amplitude of the motion. We have done some more experiments and upload the videos to YouTube. One can go to the playlist through the link embedded next. Click Here. Greater amplification factor, α, resulted in greater distortion in the video. Also increasing the frequency band of amplification introduces higher distortion.

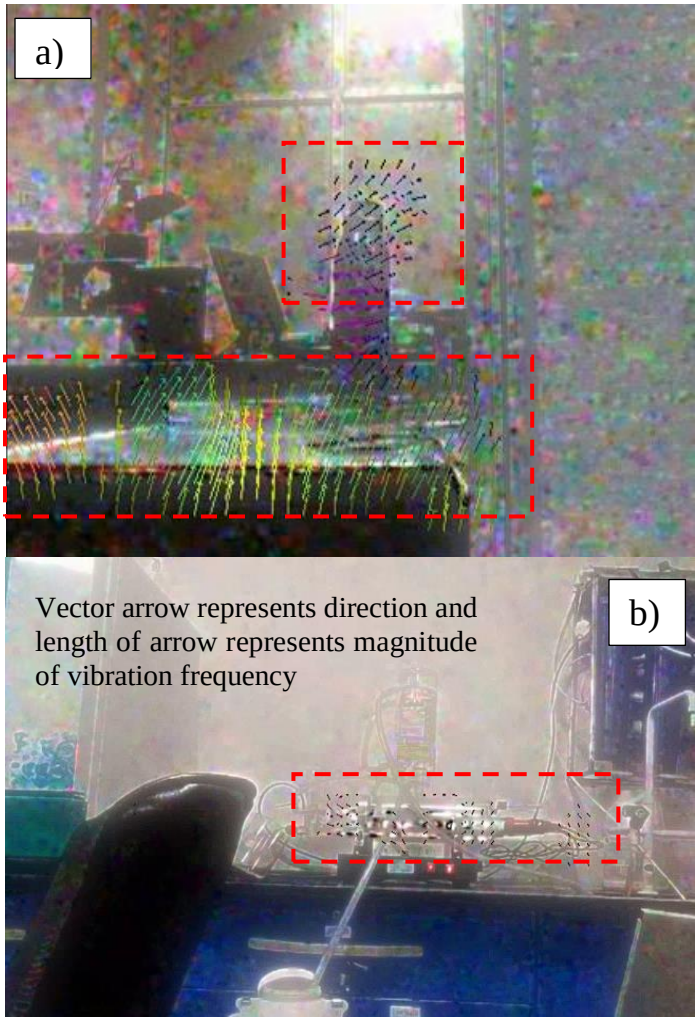

Fig.3 Optical Flow in amplified videos

The optical flow added to the video helps in visualizing the motion, its magnitude and direction clearly. Fig.3 shows two frames of different amplified videos with respective optical flow on them. Fig.3 (a) shows two moving regions, table and bottle. The table has longer optical flow vectors representing higher motion than the bottle. In Fig.3 (b) The vectors are small and mainly attached to two regions, at the boiler and at the pipe, showing the moving water in the boiler and the vibrating tube due to the motion. The amplified videos of both the above cases are uploaded at the link given above.

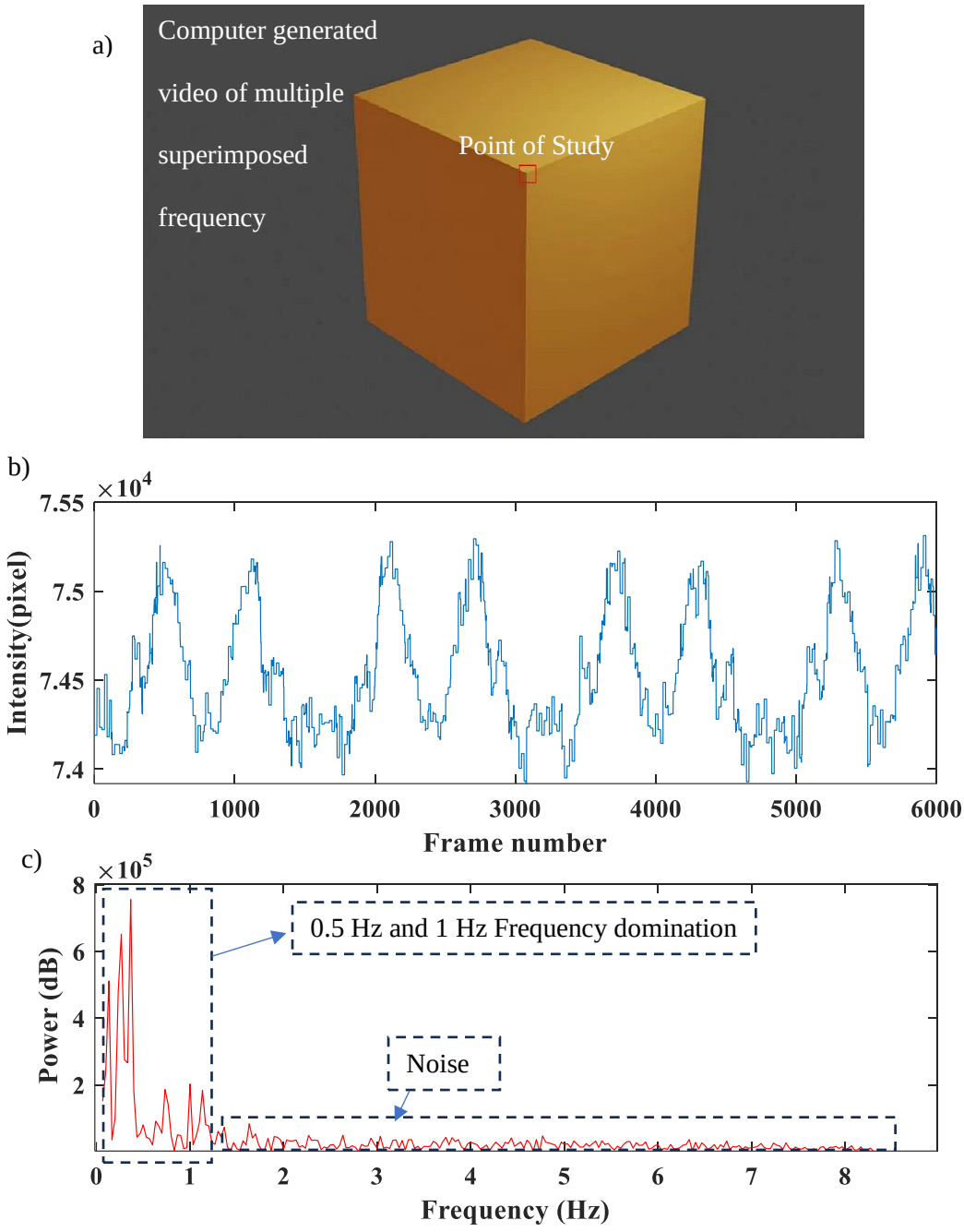

Fig.4 Frequency analysis result of a computer-generated video

For the validation of frequency analysis, we generated a video of a cube with motion in X-Y plane. The Eq.10 was used with $a_z = 0$, and the ratio of amplitude in X-direction to

that of Y-direction was 4, and frequency 1Hz and 0.5 HZ respectively. The Fig.4 (b) shows the recorded motion in the voxel shown in Fig.4 (a), this data is then passed through FFT to get the resulting waveform in the Fig.4 (c). It shows the dominant frequencies around the 0.5Hz and 1Hz, as expected.

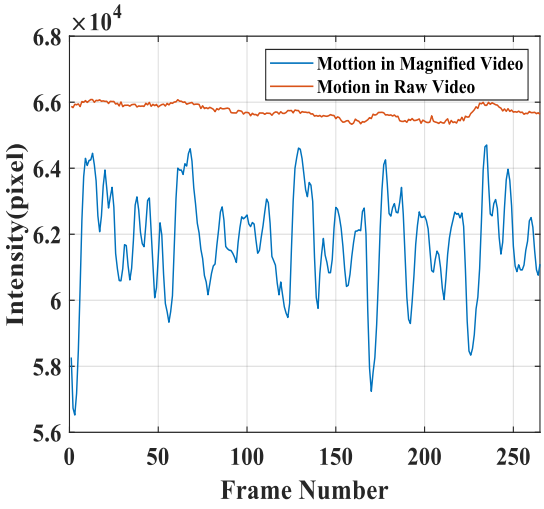

Fig.5 Motion Signal obtained from Magnified and Normal Video

The amplification process greatly improves the obtained motion signal obtained from the voxel. Fig.5 shows two signals, the motion signal in the magnified video has greater amplitude which helps to improve the frequency analysis and qualitative visualization of the motion.

7. Future Direction

While our project represents a substantial step forward in the field of VMM, there are several exciting directions for future work:

- **Hardware Improvements:** Future iterations of this project could benefit from access to higher frame rate cameras, enabling the measurement of higher frequencies. We also aim to use GPU to have faster processing.

- **Noise Reduction:** Exploring advanced noise reduction techniques and image enhancement algorithms could improve the accuracy of VMM in real-world scenarios.
- **Integration with IoT:** Integrating our software suite with IoT devices and sensors could expand its applicability, enabling real-time monitoring and analysis of various systems.
- **Machine Learning:** Incorporating machine learning techniques for motion detection and noise reduction could enhance the robustness and accuracy of the VMM process.

8. Conclusion

In conclusion, proto-typing for hackathon has allowed us to develop a powerful and accessible tool for Video Motion Magnification. While challenges and limitations exist, the software's practical significance and potential for future enhancements make it a valuable contribution to the field of vibrational analysis. We look forward to further refining and expanding our project to address real-world problems and applications.