A REPORT

ON

A STRUCTURAL HEALTH MONITORING SYSTEM VIEWABLE IN AUGMENTED REALITY

BY

Tanvi Ganu 2017A3PS1901G

AT

CEERI, Pilani

A Practice School-II station of

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

(December, 2020)

A REPORT

ON

A STRUCTURAL HEALTH MONITORING SYSTEM VIEWABLE IN AUGMENTED REALITY

BY

Tanvi Ganu 2017A3PS1901G

Prepared in partial fulfilment of the
Practice School-II Course

AT

CEERI, Pilani

A Practice School-II station of

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

(December, 2020)

Acknowledgements

I would like to express my sincere gratitude to the director of CEERI, Pilani for giving me this valuable opportunity to work in this reputed institution. Further, I would like to thank Mr. Vinod Kumar Verma sir, the incharge of the training section at CEERI. I would like to also thank Mr. Pawan Sharma sir for coordinating the entire PS-programme with this institution. Finally, I would like to express my sincere thanks to my mentor for this project, Mr. Pramod Tanwar sir, for giving me this opportunity as well as guiding me through every step of this project.

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE
PILANI (RAJASTHAN)
Practice School Division

Station: Central Electronics Engineering Institute, Pilani

Centre : Pilani

Duration: July 2020 - December 2020

Date of Start: July 24th, 2020

Date of Submission: December 16th, 2020

Title of the Project: A Structural Health Monitoring system viewable in Augmented Reality

ID No./Name(s)/ Discipline(s)/of the student(s) : Tanvi Ganu - 2017A3PS1901G, Electrical and Electronics Engineering

Name(s) and designation(s) of the expert(s): Mr. Pramod Kumar Tanwar, Principal Scientist

Name(s) of the PS Faculty: Mr. Pawan Sharma

Key Words: Structural Health Monitoring, Augmented Reality, Civil Engineering

Project Areas: Augmented Reality

Abstract: Augmented Reality is an effective and easily accessible tool to mix the virtual world with the real world environment. Structural health monitoring is viewed as a complex process which needs access to sensors and huge computational power to process data in order to get

results regarding the damage present in the structure. The purpose of this project is to reduce this workload and be able to directly access information regarding the health of the structure in an augmented form through a device screen. For this purpose, the sensors used will also be readily available and easy to access by even citizens. The user of this device should be able to visualize the damage detection present in an augmented form through the device they are holding and ideally be able to view the severity and location of the damage. This system not only makes damage detection efficient, but also helps workers and trainers in the long term maintenance of the structures in question. We will especially be focusing on the post earthquake stress on the structures in question.

Signature of Student - Tanvi Ganu
Date - 8/10/2020

Signature of PS Faculty
Date

Table of contents

Introduction

Structural Health Monitoring (SHM) refers to the process of implementing a damage detection and characterization strategy for engineering structures such as bridges and buildings. This may be done for structures that are being tested for a new design or the monitoring of the integrity of a structure during its lifetime or after a calamity strikes. There are four main stages involved in the process of SHM - 1) To detect the existence of damage in the structure 2) Locating the damage 3) Identifying the types of damage 4) Quantifying the severity of the damage.

For this monitoring, SHM utilizes data acquired from a network of sensors that may be embedded into or even simply attached to the structure. This sensor network will then consist of a data processing system for data acquisition, transmission, as well as storage. Additionally, it would also consist of a health evaluation system that analyzes this procured data.. To optimize this system, it is necessary to appropriately position an optimum number of sensors on the structure, so as to benefit from direct and full information for structural inspection and maintenance.[1]

Augmented reality is defined as “a technology that superimposes a computer-generated image on a user's view of the real world, thus providing a composite view.” It is the superimposition of virtual objects onto the real world for us to simulate an environment that does not exist. This technology enables users to take advantage of the surroundings and real-world environment as well as carry out necessary operations.

While analyzing the relation between an augmented reality enabled device and SHM, there are a few major factors that must be considered and act efficiently in order for a successful system. The factors that prove to be of utmost importance is the efficiency of being able to produce real time updated output as the user moves around while providing minimum constraints on the user’s position and orientation.

The relationship between outdoor structure analysis and Augmented Reality began as early as 1998, Lawson, Shaun W. and John RG Pretlove built an augmented reality for finding faults in

underground pipe inspection and maintenance.[2] Thereafter, in 2003, Wayne Piekarski and Bruce H. Thomas demonstrated the use of AR User Interfaces and techniques for outdoor architectural modelling using body relative and working plane techniques.[3]

In 2007, Vinnet R.Kamat and Sherif El-Tawil, M.ASCE evaluated the possibility of using augmented reality for rapid assessment of Earthquake-Induced Building damage, where it was concluded that in order to achieve a feasible and unproblematic solution, registration should be achieved by tracking user position and orientation, rather than a marker-based AR solution.[4]

As recently as 2019, an augmented reality system was developed for efficient aeronautical maintenance - which increased efficiency and reliability of tasks performed, ensuring the use of AR as an appropriate way of inspecting maintenance of any structure.[5] Moreover, in 2020, a simple strain-gauge project clearly showed that real-time data visualization by an AR headset is a feasible possibility.[6]

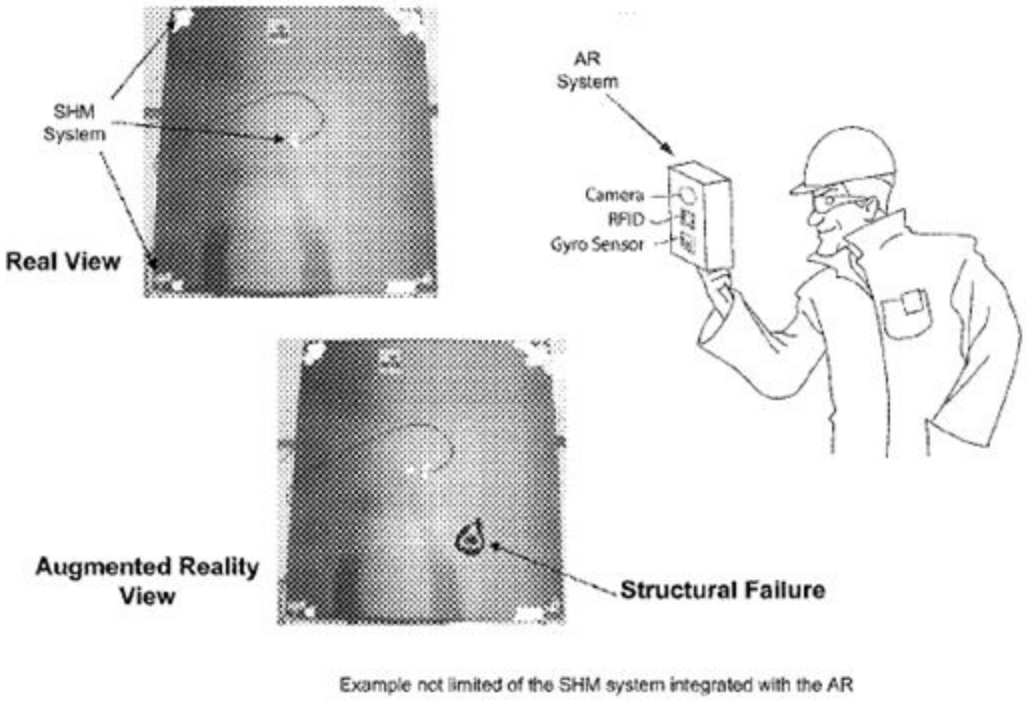


Fig 1 - A functioning Augmented Reality SHM device

The aim is to create an inspection system for detecting structural damage to a structural platform, that should be viewed through an augmented reality device. The device should track the structure as well as receive data from the sensors attached to the structure in order to monitor the health of the structure. Moreover, the AR device screen would show real world images via the camera attached to it, along with virtual indications of structural damage. This system is currently focused upon seismic activity and its damage to structures.

2. Finite element analysis in Structural Health Monitoring

2.1 Definition

Finite element analysis is the simulation of a physical phenomenon using a numerical mathematical technique referred to as 'Finite Element Method'. This method essentially breaks down any object into a large number of finite elements. Mathematical equations are used to predict the behaviour of each element, and in the end all individual behaviours are added up to predict the behaviour of the actual object. FEA is often used as a way of mathematically modelling the stresses on an engineering design. Consequently, it forms an important part of structural health monitoring of any structure.

Finite element analysis can be conducted on paper or through computer simulations via softwares. For experimental purposes, the student Ansys software was chosen to apply a constant force to a cantilever steel beam with fixed support. This gives us insight into the kind of data that can be made available through finite element analysis, mainly stress and strain plots at various locations of the beam.

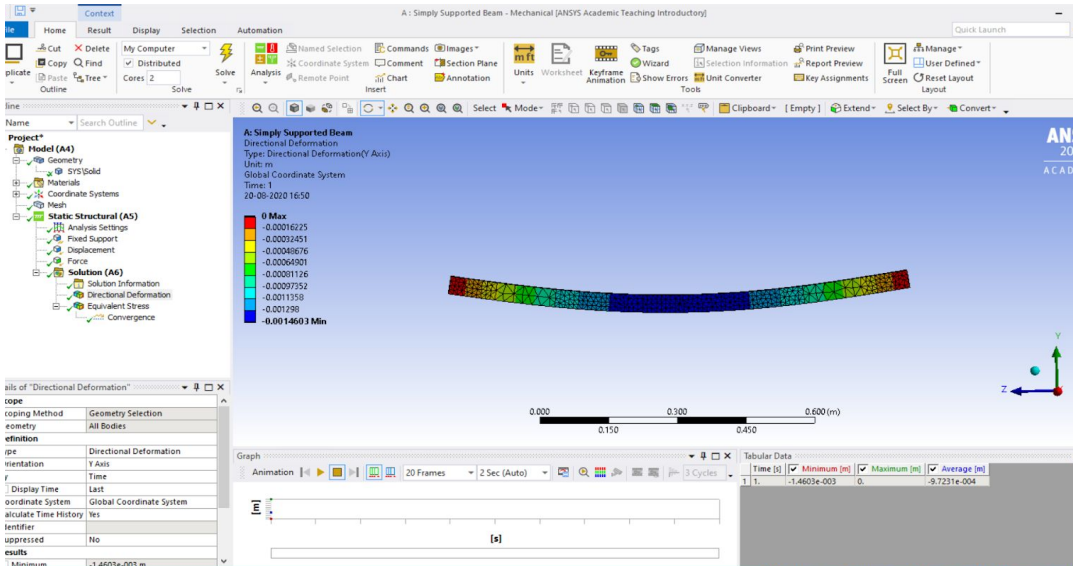


Figure 2 - Ansys software simulation of a constant force on a cantilever beam with fixed support

Apart from Ansys, there are many other paid softwares that can be used which may provide better accuracy as per the requirement of the experiment.

2.1 Simulation in Unity3D engine

2.2.1 Unity's Physics engine

Unity3D engine is a powerful tool used for the purpose of creating Augmented reality applications, amongst other functionalities. The Unity Physics Engine enables objects to approximate universal forces in nature such as gravity, velocity, acceleration, and friction. It allows for a wide range of objects with various physical properties to interact with other objects in a scene dynamically.[26]

Of the many properties that the physics engine can alter, one of the most commonly used ones for solid objects in the scene is called the 'Rigidbody property'. This property allows GameObjects to be affected by physics properties such as gravity. It also includes properties such as mass, acceleration and drag (air resistance). However, one disadvantage of Unity and the rigidbody property is its lack of properties to introduce deformations within an object due to external forces.

Physics joints is another property that can be used to introduce flexibility into the scene. However physics joints act between two rigidbodies and hence have to be attached. This property can be useful for replicating human bodies joints or rotation about a hinge. However its properties are not suitable for our current requirements of structural deformation.

2.2.2 Meshes in Unity

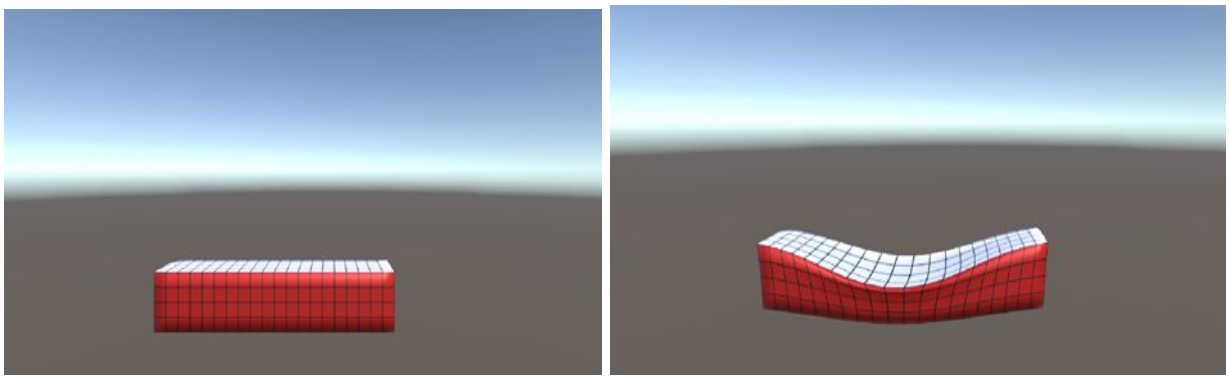
Unity does not have its own mesh analysis tool to simulate the effect a force can have on a structure. In order to recreate an environment similar to Unity, one method adopted was to reconstruct the mesh from scratch and apply physical properties to it as per our wish.

In Unity we do not have a mesh grid to alter and edit as per our requirements in order for structural deformation to occur. Hence, for this purpose the first step taken was to create a grid of

cube vertices using the “OnDrawGizmos” function in Unity. Further, once a basic cube formation using vertices is achieved, we create quadrilateral meshes linking all the vertices. It is important to note that we are reconstructing this entire mesh from scratch in order to have control over its alteration in the future. Using surface shaders and reconstructing normals, we are able to achieve a 3D structure of a beam with rounded corners.

To apply actual force to the mesh, we make a class called “MeshDeformer” to define the velocities of displaced vertices, the displacement, and so on. We can use this MeshDeformer script to define the action that takes place on application of a force. We then create a separate script for the actual application of the force. The force values may vary as per our requirements of the environment. Using the Physics raycast property of Unity, we are able to pinpoint the actual location, with the help of our cursor or touch, where the force must be applied to the mesh. The displacement of vertices then occurs as per our definition, by converting the force applied to velocity, as in the script.

One limitation of this is we are unable to specifically define material properties in Unity. As an attempt to make up for this drawback, three more variables were taken into consideration - Offset, spring force and damping. Offset refers to the offset vertices affected by the force applied at a particular location. Spring force refers to the bounce back experienced after the force has been applied. Damping refers to the speed of the application of the force on to the material for deformation. Altering these 3 properties can give us a closer simulation to the material properties of our choice.[7]
Using these properties in C# scripts helped achieve a near simulation as observed in Ansys software - (given in Appendix A)



(a) (b)

Fig 3 - (a) Original beam structure as constructed in Unity (b) Deformed beam structure when a force of 10N is applied to centre of beam

This simulation shows us that mesh deformation can create an Ansys like environment in Unity. The limitations to the current simulation are to replicate the exact material properties as we are able to in Ansys. This would require complex physics formulae and additional variables to our current system, such as Young's modulus, stiffness coefficient etc.

As of now, this system has not been further explored.

3. Seismic Vibration detection and simulation

Amongst the many natural disasters one can simulate in order to test a damage detection software, one of the easiest to produce and simulate is that of seismic activity during an earthquake.

3.1 Data collection for vibration analysis

Typically for detection of seismic vibrations and frequency data, highly accurate and sensitive piezoelectric sensors are used. However, access to these sensors and its data is not easily available. For easily available and collectible data, the search was for a more worldwide spread device.

Accelerometers are devices that measure acceleration. These devices are generally present in most new generation smartphones on the market today. Accelerometers have the ability to detect any change in orientation of the smartphone device as well. This can prove advantageous to us in many respects.

In 2015, in an attempt to begin a research on CitizenSensor for SHM, frequency data from smartphone accelerometers was directly compared with highly accurate and sensitive measuring instruments, and it was found that the results were very comparable.[8] This proved a very vital point in the entire experiment, that smartphone accelerometers can be used to detect data and change in frequency.

The next task was to integrate this data in Unity software. In order to access the acceleration data, we use a function that treats acceleration of the device as an input. Taking the value of this input as a vector in X,Y and Z axes, we are able to find the change of acceleration of the device in fixed frame updates (taken as 1 second in this case). We are also able to immediately display the measured quantities, along with timestamps in Unity. This acceleration data is measured in the standard unit of metres per seconds squared. Using this data, we can also plot acceleration vs time graphs, integrate and find velocity and displacement vs time graphs. We can hence find the frequency of the given movement through this simulation.

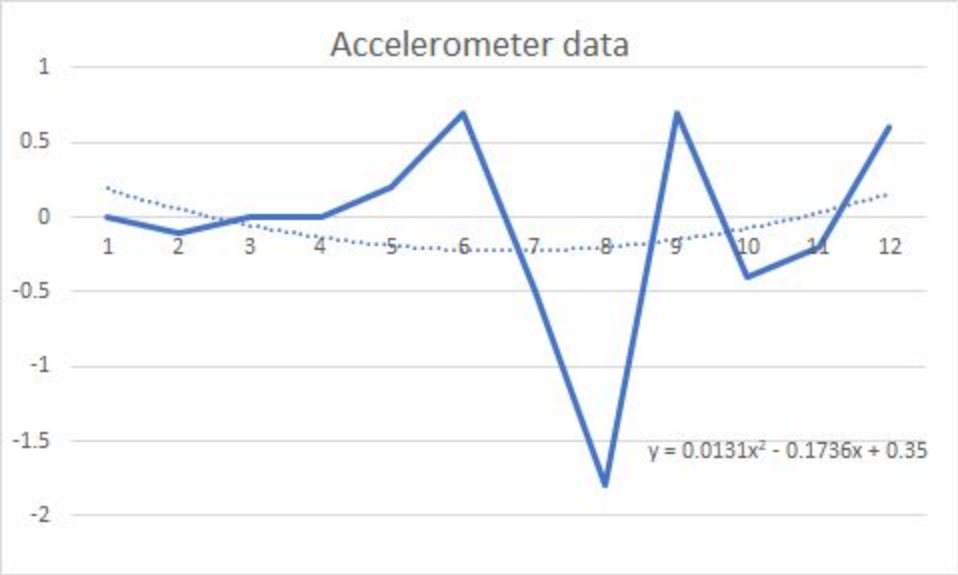


Fig 4 - Acceleration graph plotted from basic oscillatory motion experiment

Additionally, we are able to use this accelerometer data to simulate the required forces onto any asset or object in Augmented Reality. With knowledge of mass of the object, we can simulate the forces in the exact axes of sense vibrations by the device. This will help us in simulating an exact replica of the real world environment into Unity.

3.2 Enabling flexibility in structure for vibration in Unity

A cantilever beam, when vibrating at its fundamental frequency or in some harmonic, clearly displays a wave-like formation in its vibrations. When any structure experiences forced vibrations, it would vibrate at that frequency in a similar manner.

The first step to this approach is to simulate these vibrations and harmonics in a cantilever beam, given its fundamental frequency, and if possible, length of beam.

For this purpose, we first construct a beam in Blender3D software with a bone structure that allows movement of multiple segments within a structured object. These bone structures are created using ‘armatures’ in Blender.

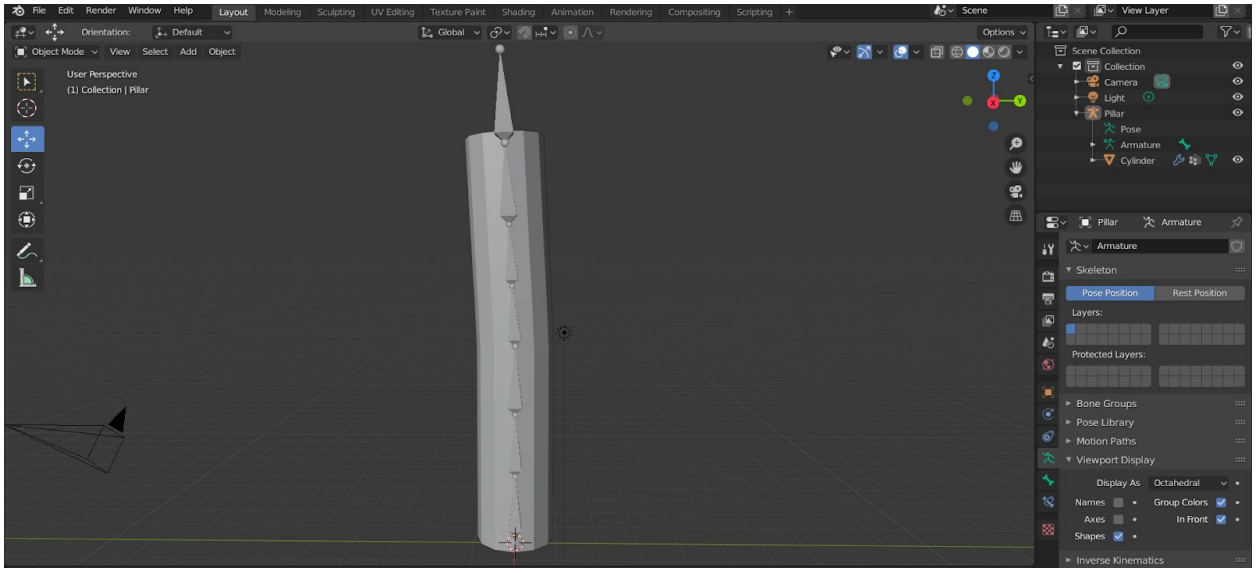


Fig 5 - Bone structure in a vertical pillar made in Blender3D software

Bones are the base elements of armatures. Bones in armature can be classified into two types - Deforming and Control bones. Deforming bones directly influence the positions of the vertices they are associated with, when they transform. Control bones, on the other hand, act as a switch and control how other bones or objects react when they are transformed. It is important to note that there are different types of armatures available in Blender, that must be chosen as per the requirements of the structure's flexibility, material and purpose.

For example, Bendy Bones (B-Bones) are an easy way to replace long chains of many small rigid bones. It is generally used for spine columns or facial muscles. For our purpose, we are using it to establish extreme flexibility in the structure.

Blender treats the bone as a section of a Bézier curve passing through the bones' joints.[24] Each of the segments thus introduced into the model because of the bendy bones will bend and roll to follow this invisible curve representing a tessellated point of the Bézier curve. There are control points at each end of the curve that would act as the end-point of the bone. The shape of the B-Bones can be controlled using a series of properties. However, it is also possible to alter their shape indirectly through the neighboring bones (i.e. first child and parent). The properties

construct handles on either end of the bone to control the curvature.

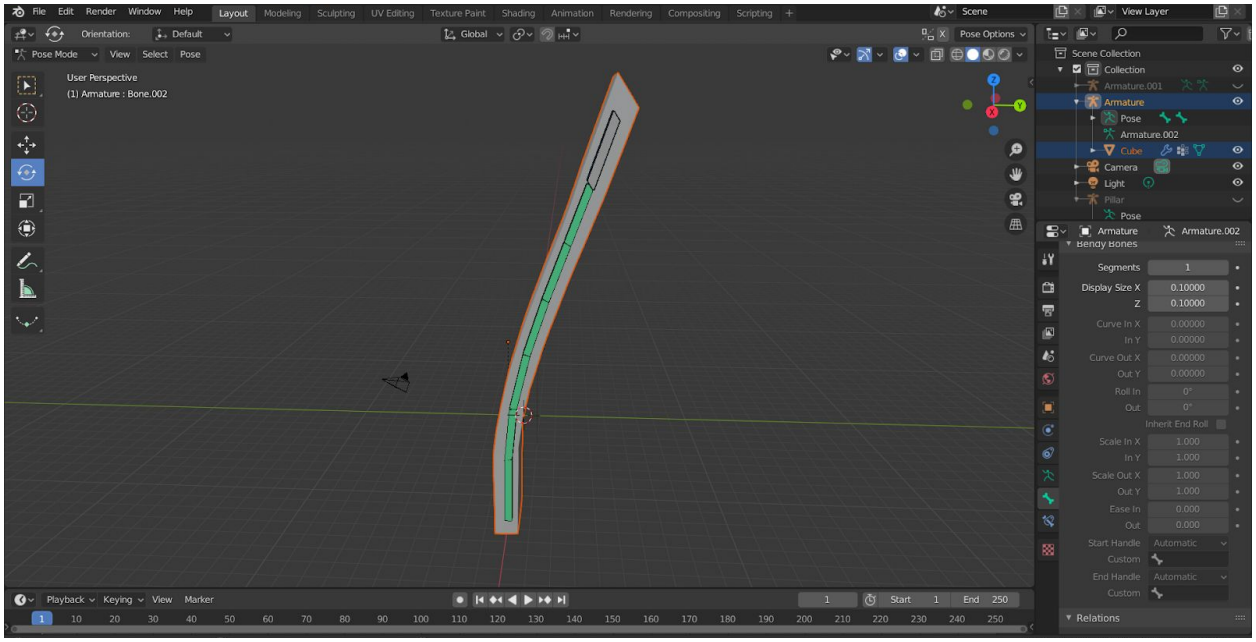


Fig 6 - Flexibility in structure has been introduced using 'Bendy Bones' in Blender

Once these segments have been imported into Unity, we can create user input for fundamental frequency and length in order to create appropriate wave-like simulation. Each segment would have to simulate an individual portion of the wave like vibrations we see. The challenge would be to enable this for a range of frequencies and harmonics that the structure may potentially experience. Once this is achieved, the advanced steps would be to include resonance as a concept.

3.3 Analysis of data acquired

SHM implies the breaking down of a structure into multiple segments and individual analysis of each segment to produce an all inclusive result of the entire structure. When it comes to the frequency data we receive, it is not for all 100+ segments that may be present in the structure. Hence we find ourselves with less measurement data as compared to the number of segments in question.

For this purpose we use the l1 regularization method in Machine Learning. When input features have weights closer to zero - that leads to sparse L1 norm. In Sparse solution, majority of the

input features have zero weights and very few features have non zero weights. In L1 regularization we penalize the absolute value of the weights.

Using this method, the number of measured data required is considerably smaller than the number of structural elements.The advantage of using frequency changes is that frequencies can be measured accurately and inexpensively. In 2015, a study was conducted to use l1 regularization on acquired frequency data and directly compare it with the l2 regularization method output. It was found that the numerical examples verified that the proposed technique of l1 regularization can successfully identify moderate damage, whereas the conventional l2 regularization cannot.[9]

This gives us further insight into how to handle that data that we acquire from the sensors. It is also advantageous as structural damage only occurs only in specific areas of an otherwise large structure. They exhibit stiffness reduction and hence can be detected from frequency response data of the structure.

4. Wind-load effect on structures

4.1 Definition

Wind produces three kinds of effects on tall buildings or structures - static, dynamic and aerodynamic.[10] It is observed that when the building is flexible, it interacts with the wind load and affects the response - also known as the aerodynamic effect. The dynamic response of tall buildings is dependent on multiple factors such as structural stiffness, mass, damping and architectural shape and form.

Natural frequency, also known as eigenfrequency, is the frequency at which a system tends to oscillate in the absence of any driving or damping force. If the oscillating system is driven by an external force at the frequency at which the amplitude of its motion is greatest (close to a natural frequency of the system), this frequency is called resonant frequency.

The estimation of wind-load and the reliability of wind-excited structures have been investigated extensively during the last few decades. Any structure exposed to a fluid stream such as wind, is subjected to an harmonically varying force in a direction perpendicular to the stream.[11] Tall structures such as masts, bridges, and chimneys are susceptible to excitation from steady winds blowing across them. This excitation may lead to a concept known as mechanical resonance, which is the tendency of a system to respond at a greater amplitude when the frequency of its oscillations matches the system's natural frequency of vibration than it does at other frequencies. The aim is to create a visually perceivable system to detect the effect of wind-loads on civil structures. The visual perception would arise from the Augmented Reality depiction of the structure and any effect the wind of varying speeds would have on it. The application would be dependent on factors such as the material of the structure, the speed and nature of the wind and the terrain in which the structure is present. Analyzing this effect would give the user greater insight into resonant frequency effects and data, for the use of maintenance and future planning.

4.2 Literature review

Wind-induced vibrations on all kinds of structures, especially tall ones, has been an important concern for many years now. The effects of wind-load is largely dependent on a number of variables which keep changing, from region to region as well as the time of the year. However, there has been a significant amount of research done in the possibilities of any kind of damage due to hazardous wind speeds in the region.

In 2003, there were full-scale simulations of wind-effects conducted under typhoon conditions on multi-storeyed buildings to provide a basis for further study into the dynamic responses of the building.[12] There has also been case-specific research and analysis of structures post a catastrophic event such as a tycoon, for example - the wind hazards in East Asia from 2013 to 2016.[13] It is also important to note wind-load dynamics in early-stages of the design process with an intent to give indication of the dynamic properties of the building.[14] IIT Kanpur provided an in-depth commentary on the wind-loads on buildings and tall structures, highlighting the variation of speeds and terrain in different regions of India.[15]

4.3 Methodology

4.3.1 Physical Properties

For the purpose of simulation, it is necessary to replicate fairly accurate material properties on the software. We can do so by constructing a steel cantilever beam in Blender using the Blender Physics engine, using the 'calculate mass' property using the density of steel as 7860 kg/m^3 and defining the dimensions of the beam.

In order to define the rigidbody properties of steel in Blender, we must interpolate the values with the given properties in Blender.

For the modulus of elasticity (E) of steel (~200 GPa), where we can approximately assume the E of Rubber as the minimum value (~0.01-0.1 GPa)[16] and maximum as the E of Diamond (~1220 GPa) - we use the bounciness (range from 0-1) property in Blender. Here 0 is a perfectly rigid body and 1 is assigned to a perfectly elastic body.[17] Accordingly, bounciness can be defined as

0.836. Additionally, we can define the coefficient of friction of rough steel with soil to be 0.5~0.6.[18]

4.3.2 Oscillations in Blender

One of the options to achieve oscillations at a particular frequency, is to use the Blender software's animation feature. As our structure has been created in Blender itself, it is useful for us to create animations in the same software.

In order to enable constant frequency oscillations, we can create a shape-key that defines the deflection in the top of the beam in the Y-axis. Hence, the extreme values of the shape-keys define the amplitude of the oscillation. Now, to enable a sine-wave function modifier into the beam's movement, we use the F-curve modifier in Blender's animation graph editor. This enables us to oscillate the beam in a sine-wave at a particular frequency, in this case known as a phase multiplier.

For the calculation of oscillations in Blender (from hertz to phase multiplier units), we observe that the original F-curve with a built-in function was found to roughly repeat itself every 6.283185307 units.[23] Hence, depending on the frequency of the required oscillation, we can multiply the appropriate value to this number.

For example, if the required frequency of oscillation is found to be 0.88 Hz, we can estimate the phase multiplier to be -

$$Phase\ multiplier\ =\ 0.88\ \times 6.283185307\ =\ 5.5292028\ units$$

This value can be set in Blender's F-Curve modifier properties, and hence the required oscillation can be observed.

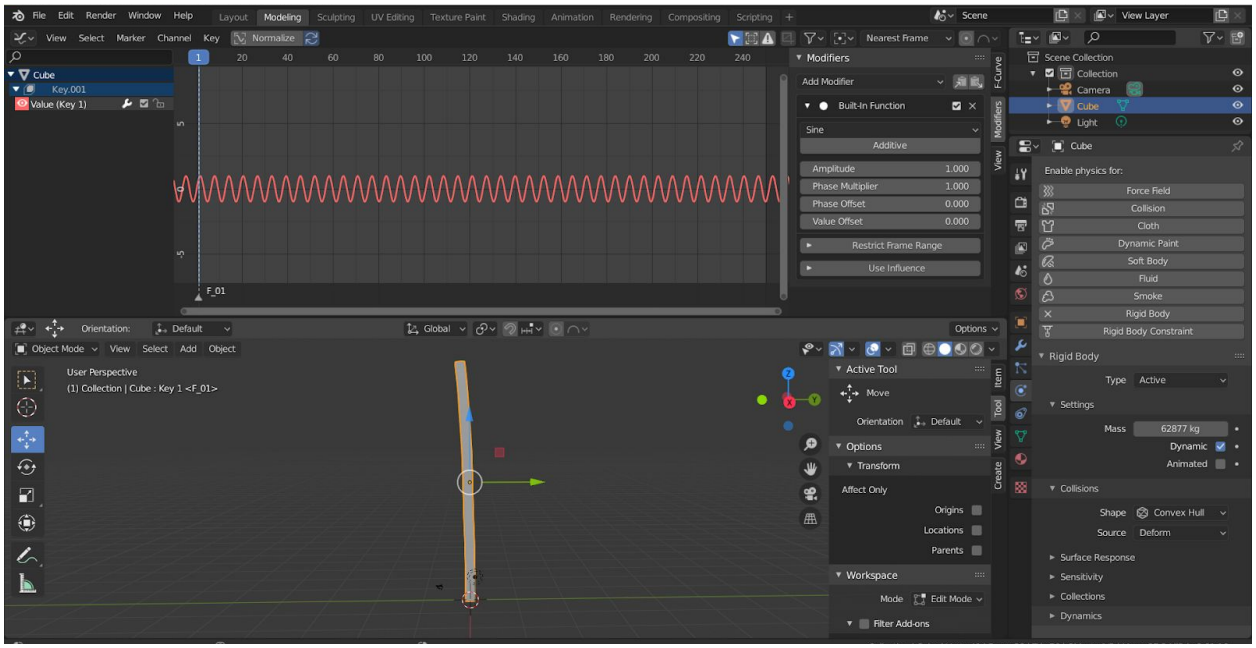


Fig 7 - Modifying oscillating movement of Pillar in Blender using F-curve modifiers

The disadvantage of this method is that though the oscillations observed, along with the density, mass, height and elasticity properties are accurate, the physics engines of Blender and Unity have been found to be incompatible. Hence, the shape keys of the structure translate into Unity, however the animation and oscillation sequence of F-curve properties, as well as mass and density properties do not.

We thus are forced to create oscillations using Unity's animation feature itself, if we wish to view the structure in Augmented Reality.

4.4 Velocity of wind

Wind load value depends on roof geometry, wind exposure, location, and its importance.[19] It also depends on the topography of the surrounding area. Taking all these variables into consideration, we can find the base speed of the wind to be considered for further calculations.

Wind velocity is found to increase with increasing height above ground level.[15] We can start off by finding the basic wind speed in that area of the country. According to long-term hourly data collected from 70 meteorological centres of India Meteorological Department, the basic wind

speed in the area under consideration, that is Pilani, Rajasthan, is found to be 47 m/s.[20] We can consider this as our V_b.

The theory behind the varying wind-speeds with height has been proposed and drafted in a commentary in 2002 [15], and we can directly apply the formulae and conditions to our current use-case. According to their commentary, the revised wind speed for tall structures can be found to vary on the basis of three parameters - risk coefficient (R), terrain, height and size factor (K) and topography factor. The revised wind-speed can be found using the formula -

$$V_z = K \,.\, R \,.\, V_b$$

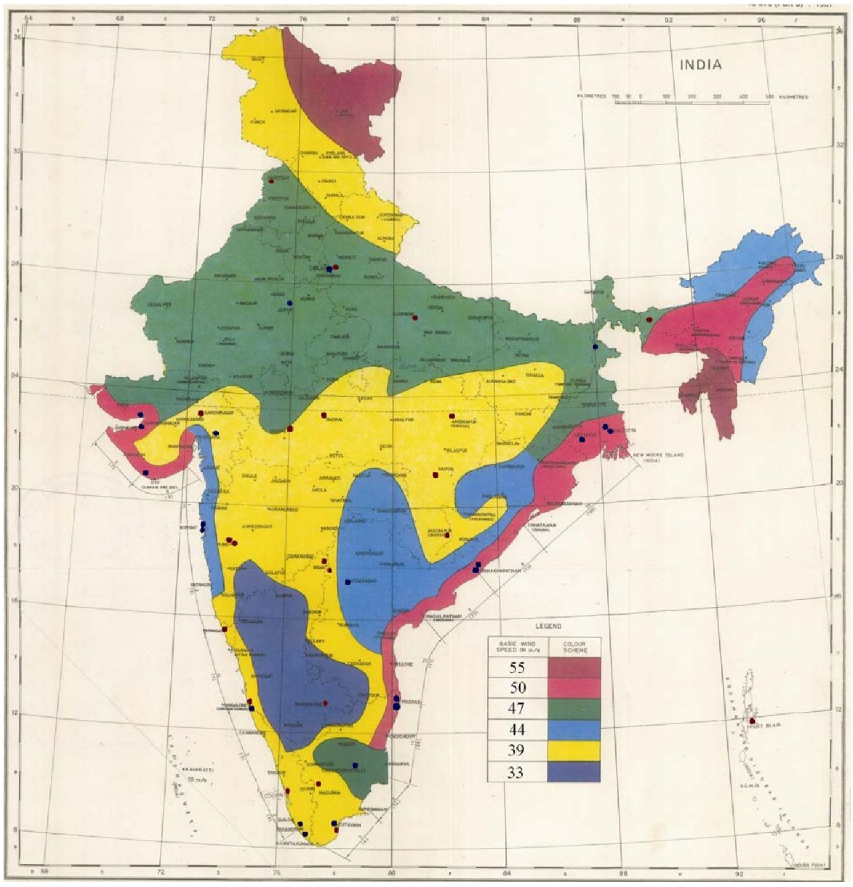


Fig 8 - Basic wind-speed map of India with long term hourly wind data

Considering the parameters of the location of the structure, with the base wind-speed found to be 47 m/s, we can estimate a risk coefficient of 1.07. Considering the surrounding area of the structure under consideration, we consider our location to be with numerous closely spaced obstructions having the size of building-structures up to 10 m in height with or without a few isolated tall structures. Studies showed that upstream buildings of the same height reduced the wall pressures and the pressures at the leading edge of the roof significantly, but had less effect on pressures on other parts of the roof. The building height/spacing ratio was the major parameter, with the number of shielding rows being of lesser importance.[21] Hence, from the 4 categories specified, we consider a terrain category of 3. Using this knowledge, and the height of the structure, we can find the value of K. We can hence arrive at a conclusion for the base wind-speed at the highest point of the structure.

4.4.1 Calculating force based on wind-speed

Calculating the force of wind based on its speed requires knowledge of the approximate mass of the air and acceleration of the wind. The average density of a mass of air at sea level equals approximately 1.229 kilograms per cubic meter.[25]

Using Newton's laws of motion, we know

$$F = m \times a$$

We assume the force of the wind in this case is impacting an area of 1 m^2. Hence, for acceleration we must now convert the speed from km/hr to m/s. For example, a speed of 10 km/hr would be -

$$1\ km/hr = 0.277778\ m/s$$

$$Hence,\ 10\ km/hr = 2.77778\ m/s \approx 2.78\ m/s$$

Thus, $F = 1\ m^2 \times 1.229\ kg/m^3 \times (2.78\ m/s)^2$

$$F = 9.4982\ kg - m/s^2 \approx 9.5\ N$$

4.4.2 Deflection at the free end

The maximum deflection at the free end of the cantilever beam can be expressed as[26] -

$$\delta = \frac{FL^3}{3EI}$$

where - δ is maximum deflection

E is modulus of elasticity

I is moment of inertia (m^4)

F is single acting force at the free end

Taking the example of the force found due to wind speed above, we can estimate deflection at the free end to be

$$\delta = \frac{9.5 \times (50)^3}{3 \times 200 \times 10^9 \times 0.02}$$

$$\delta = 9.895 \times 10^{-5}\ \text{m}$$

4.5 Natural and Resonant frequency

4.5.1 Natural frequency of a Cantilever beam

The natural frequency (N) of the structure can be found using the following formula used for a cantilever beam of length L[22] (Appendix A -(1)),

$$N = \frac{1}{2\Pi}\sqrt{\frac{3EI}{mL^3}}$$

E = modulus of elasticity

I = second moment of inertia of area of the section

m = mass undergoing vibration

For example, let us consider the height of the structure to be 50 metres. Hence,

$$l^3 = (50)^3 = 125000$$

The density of steel as registered in Blender is 7860 kg/m^3. Hence we find the mass of the structure to be 62,877 kg. Further, the modulus of elasticity can be found to be ~200 GPa and I can be estimated to be 0.02 m^4 [(11)]. Thus the natural frequency can be found as -

$$N = \frac{1}{2\Pi}\sqrt{\frac{3\times 10^9 \times 200 \times 0.02}{62877 \times (50)^3}}$$

$$N = 0.196657\ Hz$$

Thus, we find the natural frequency to be approximately 0.196 Hz.

With the given properties we would be able to find the natural frequency of the structure in question. Our next concern would then become of the vibration of the buildings reaching the resonant frequency due to the wind-load.

4.5.2 Resonant frequency

As a previously worked upon problem[11], for a cantilever beam with a deflection of δ at its free end, the angular frequency of vibrations can be found out using the following equation

$$\omega^2 = \frac{EI\delta^2(\frac{\Pi}{2l})^4 \frac{l}{2}}{\delta^2 \frac{m}{l}(\frac{3}{2}-\frac{4}{\Pi})l}$$

Finding the value of ω , we can find the value of f, and hence substitute it in the Strouhal equation to find the frequency at which the beam will oscillate (f_s).

This equation tells us that the angular frequency at which resonance will occur is independent of the deflection at the free end. Simplifying the equation and substituting values of m = 62,877 kg, l = 50 m, E = 200 GPa and I = 0.02 m^4 we get -

$$\omega^2 = \frac{200\times 10^9 \times 0.02 \times \Pi^4}{32l^3 \times m \times 1.27324}$$

$$\omega^2 \simeq \frac{389.6363}{320}$$

$$\omega^2 = 1.217613\ rad/s$$

$$\text{Hence, } \omega = 1.129659\ rad/s$$

Hence the frequency would be

$$f = \frac{\omega}{2\Pi} \simeq 0.179\ Hz$$

In dimensional analysis, the Strouhal number is a dimensionless number describing oscillating flow mechanisms. It can be found with the knowledge of frequency of flexural vibrations (f_s), diameter (D) and velocity of current(v). Experimental evidence suggests a value of 0.20-0.24 for Strouhal number for most flow rates and wind speeds encountered.[11]

$$\frac{f_s D}{v} = 0.20\sim0.24$$

Using the above calculations to determine velocity of the wind at which oscillations of 0.179 Hz will occur, and assuming a Strouhal number of 0.22 with a diameter of 5 m,

$$v = \frac{0.179 \times 5}{0.22} = 4.068\ m/s = 14.6448\ km/hr$$

We can hence find that for the parameters we have considered, at the speed of 14.6448 km/hr, resonance will occur for this cantilever beam.

Hence, knowledge of the speed at which resonance will occur can be directly compared with our knowledge of wind speeds in the area.

4.6 Simulations in Unity

4.6.1 Simulation of wind particles

Using Unity's particle system, we can modify the speed of the wind seen in order to generate realistic effects in the User Interface. This can be achieved using the Particle System's 'Velocity over Lifetime' property.

Depending on the choice made in the dropdown for wind effects, we are able to alter the 'Speed Modifier' property, and hence accelerate or decelerate the particles being bombarded at the structure.

We have 3 ranges of speed in the given UI - we can categorize as slow, medium and fast. Accordingly, we can vary the speed modifier as 1, 2 or 3. This feature enables the user to visualize the speed of the wind in Augmented Reality and hence enhances the overall effect.

4.6.2 Calculations in Unity

For the purpose of simulation, the Unity software is chosen to enable Augmented Reality features in our application. A cantilever beam of the appropriate height and width is chosen. There is a User Interface created to induce different wind-speeds of the region that would exert a lateral load on the structure. The variation of velocity of the wind with height is also taken into consideration.

Depending on the variables chosen, we will have the knowledge of the natural frequency, frequency of vibrations, as well as resonance frequency of the structure. Based on this knowledge we can display appropriate messages, issue warnings, as well as provide extra information on prevention of any disaster.

A user interface is created, enabling user input for quantities such as - wind speed, height and material. Based on this data we and the above formulae, we should be able to calculate the required data, as well as compare natural and resonant frequencies.

The range of the height is given to roughly estimate the height of the structure required to calculate the further quantities. This field cannot be left blank as there is no default value.

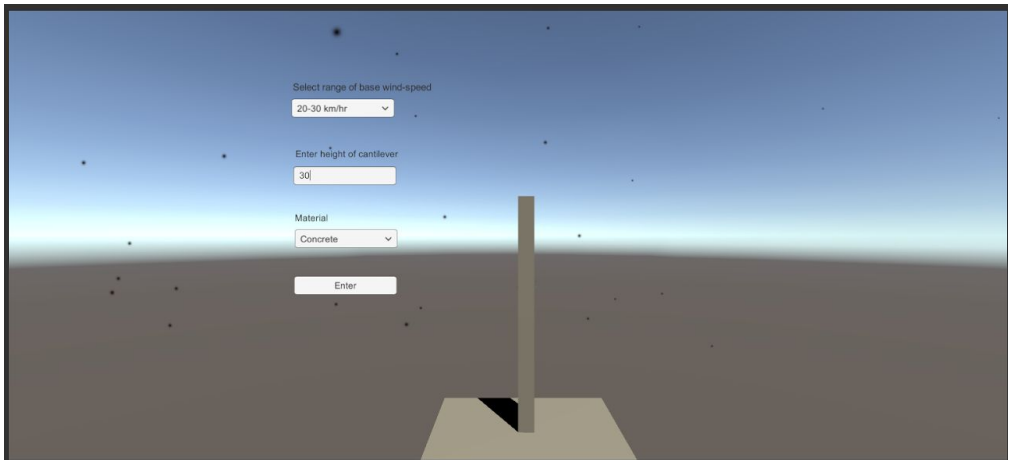


Fig 9(a) - A look at the user interface of the final product

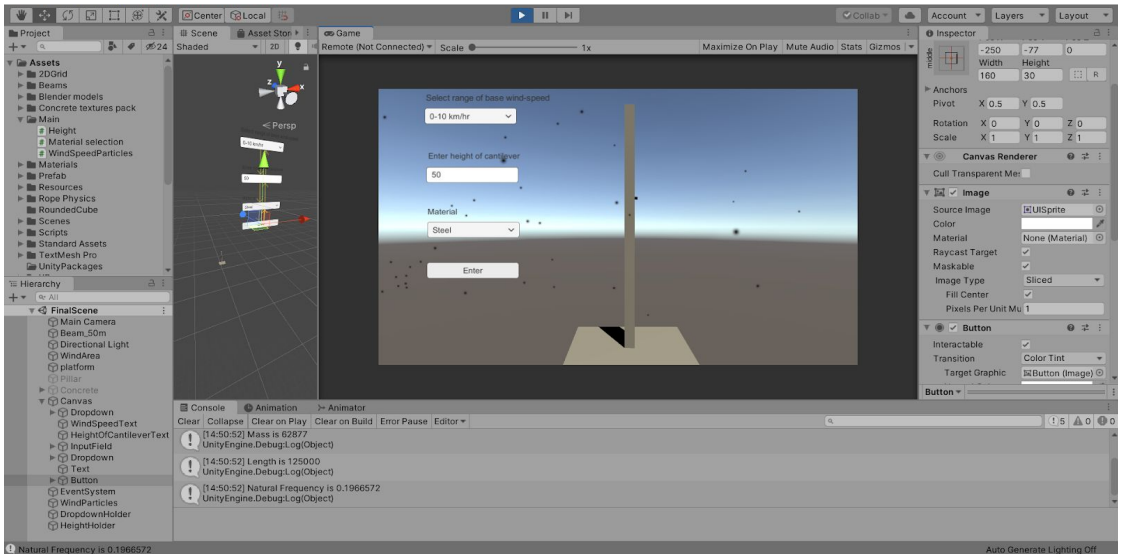


Fig 9(b) - Input parameters dependent results with mass and length calculator

For the purpose of simulation, we make use of C# scripts in order to find the natural frequencies of the structure in view, based on the user input.

We will make use of the wind speed to find the vibrational frequency, compare it with the resonant frequency, and hence find the point at which the structure will face a threat due to the effect of strong winds in the area.

The height of the structure will directly affect the wind-speed at the highest point, as it varies with height increment, as well as structural stability.

The material of the structure will directly affect the mass of the structure, modulus of elasticity, second moment of inertia, as well as the nature of the structure we are taking into consideration. The options available for this application are steel and RC concrete.

The codes used for this process can be seen in Appendix A (2)

Software specifications and limitations

It is important to note that the current progress is based on the current software versions available, which are Blender 2.8a and Unity 2019.3.13. All plugins and properties available with these software have been explored for possible application into the project. There are features, however, that cannot keep up with the requirements of the project, at least with the current version made available to the public. There is a need for a more finely made physics engine if projects like these are to be made available in Augmented or Virtual Reality. Similarly, all physics engines used and experimented with, such as Ansys Software, Google Sketchup and CAD Viewer were used with their free versions only. Hence the limitations offered by the solutions developed are due to feature limitations, which may be available in the paid version for professional use.

Further, all code has been written in C# language using Visual Studio 2019 with a Unity Debugger. Minor code snippets that have been written for the User Interface have not been included in the Appendix.

Conclusion

Structural Health Monitoring is a very vast concept involving many factors to be taken into consideration. Having focused on one or two of these factors, of Accelerometer data and wind load effects in Unity, there was much progress done in introducing core physics concepts into Unity and Blender softwares. Enabling these features in software is very important for future uses, for them to be easily viewable, altered and worked upon in the future. In this respect, this report has provided a strong base that can easily be worked upon in the future.

In this report, there has been in-depth research into the physics engine of Unity and Blender software. Moreover, there have been multiple trial and error evaluations of the possibilities and properties available to us with the present versions of these software. Further updates could lead to better and more accurate results for these experiments. There have also been separate trials conducted for sole Blender software, as the importance of Unity is only established with its use to be viewable in Augmented Reality.

In the future, sensor data from actual structures can be used for most efficient output. Taking proper dimensions into consideration, along with terrain features - accurate real-time data could be usable for helping workers, and maintenance staff for existing buildings, as well as efficiently planning future buildings with appropriate materials.

Appendix A

1) C# code for mesh deformation in Unity :

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class MeshDeformer : MonoBehaviour
{
    Mesh deformingMesh;
    Vector3[] originalVertices, displacedVertices;
    Vector3[] vertexVelocities;
    public float forceOffset = 0.1f;
    public float springForce = 20f;
    public float damping = 5f;
    public float force = 10f;
    float uniformScale = 1f;
    void Start()
    {
        deformingMesh = GetComponent<MeshFilter>().mesh;
        originalVertices = deformingMesh.vertices;
        displacedVertices = new
Vector3[originalVertices.Length];
        for (int i = 0; i < originalVertices.Length; i++)
        {
            displacedVertices[i] = originalVertices[i];
        }
        vertexVelocities = new Vector3[originalVertices.Length];
    }

    void Update()
    {
        uniformScale = transform.localScale.x;
        for (int i = 0; i < displacedVertices.Length; i++)
        {
            UpdateVertex(i);
        }
```

```
        deformingMesh.vertices = displacedVertices;
        deformingMesh.RecalculateNormals();
        if (Input.GetMouseButton(0))
        {
            HandleInput();
        }
    }
    void UpdateVertex(int i)
    {
        Vector3 velocity = vertexVelocities[i];
        Vector3 displacement = displacedVertices[i] -
originalVertices[i];
        displacement *= uniformScale;
        velocity -= displacement * springForce * Time.deltaTime;
        velocity *= 1f - damping * Time.deltaTime;
        vertexVelocities[i] = velocity;
        displacedVertices[i] += velocity * (Time.deltaTime /
uniformScale);
    }

    void HandleInput()
    {
        Ray inputRay =
Camera.main.ScreenPointToRay(Input.mousePosition);
        RaycastHit hit;

        if (Physics.Raycast(inputRay, out hit))
        {
            MeshDeformer deformer =
hit.collider.GetComponent<MeshDeformer>();
            if (deformer)
            {
                Vector3 point = hit.point;
                point += hit.normal * forceOffset;
                deformer.AddDeformingForce(point, force);
            }
        }
```

```
    }

    public void AddDeformingForce(Vector3 point, float force)
    {
        point = transform.InverseTransformPoint(point);
        for (int i = 0; i < displacedVertices.Length; i++)
        {
            AddForceToVertex(i, point, force);
        }
        Debug.DrawLine(Camera.main.transform.position, point);
    }
    void AddForceToVertex(int i, Vector3 point, float force)
    {
        Vector3 pointToVertex = displacedVertices[i] - point;
        pointToVertex *= uniformScale;
        float attenuatedForce = force / (1f +
pointToVertex.sqrMagnitude);
        float velocity = attenuatedForce * Time.deltaTime;
        vertexVelocities[i] += pointToVertex.normalized *
velocity;
    }
}
```

2) Code used for calculation of natural frequency

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Height : MonoBehaviour
{
    public string height;
    public GameObject inputField;
    public GameObject Cantilever;
    public GameObject platform;
    float heightValue, naturalFrequency;
    float mass;


    public void getHeight(string newText)
    {
        heightValue = float.Parse(newText);
        Debug.Log("Height is " + heightValue);
        Cantilever.transform.localScale = new
Vector3(Cantilever.transform.localScale.x,
Cantilever.transform.localScale.y, heightValue * 50);
        AdjustLocation();
    }

    public void AdjustLocation()
    {
        float temp = platform.transform.localPosition.y;
        Cantilever.transform.localPosition = new
Vector3(Cantilever.transform.localPosition.x, temp + heightValue
/ 2, Cantilever.transform.localPosition.z);

    }

    public void HandleMaterialInput(int val)
    {
        if(val == 0)
```

```
        {
            Debug.Log("Steel Selected");

        }
        if(val == 1)
        {
            Debug.Log("Concrete Selected");
        }
    }

    public void OnButtonPressed()
    {
        //mass = 7860 * 5 * 4 * heightValue;
        mass = 62877;
        Debug.Log("Mass is " + mass);
        var length = Mathf.Pow(heightValue, 3);
        Debug.Log("Length is " + length);
        naturalFrequency =  Mathf.Sqrt((3 * 200 * Mathf.Pow(10,
9) * 2) / (100 * mass * length)) / (2 * Mathf.PI);
        Debug.Log("Natural Frequency is " + naturalFrequency);
    }
}
```

References

1. Yi, Ting-Hua, and Hong-Nan Li. "Methodology developments in sensor placement for health monitoring of civil infrastructures." *International Journal of Distributed Sensor Networks* 8.8 (2012): 612726.
2. Lawson, Shaun W., and John RG Pretlove. "Augmented reality for underground pipe inspection and maintenance." *Telemanipulator and Telepresence Technologies V*. Vol. 3524. International Society for Optics and Photonics, 1998.
3. Piekarski, Wayne, and Bruce H. Thomas. "Augmented reality user interfaces and techniques for outdoor modelling." *Proceedings of the 2003 symposium on Interactive 3D graphics*. 2003
4. Kamat, Vineet R., and Sherif El-Tawil. "Evaluation of augmented reality for rapid assessment of earthquake-induced building damage." *Journal of computing in civil engineering* 21.5 (2007): 303-310.
5. Ceruti, Alessandro, et al. "Maintenance in aeronautics in an Industry 4.0 context: The role of Augmented Reality and Additive Manufacturing." *Journal of Computational Design and Engineering* 6.4 (2019): 516-526.
6. Aguero, Marlon, et al. "Design and Implementation of a Connection between Augmented Reality and Sensors." *Robotics* 9.1 (2020): 3.
7. https://catlikecoding.com/unity/tutorials/mesh-deformation/
8. Feng, Maria, et al. "Citizen sensors for SHM: Use of accelerometer data from smartphones." *Sensors* 15.2 (2015): 2980-2998.
9. Zhou, Xiao-Qing, Yong Xia, and Shun Weng. "L1 regularization approach to structural damage detection using frequency data." *Structural health monitoring* 14.6 (2015): 571-582.
10. Alaghmandan, M., and M. Elnimeiri. "Reducing impact of wind on tall buildings through design and aerodynamic modifications." *AEI conference, Penn State university. College Station, Penn*. 2013.
11. Beards, C. *Engineering vibration analysis with application to control systems*. Elsevier, 1995.
12. Li, Q. S., et al. "The effect of amplitude-dependent damping on wind-induced vibrations of a super tall building." *Journal of Wind Engineering and Industrial Aerodynamics* 91.9 (2003): 1175-1198.
13. Yang, Qingshan, et al. "Damage to buildings and structures due to recent devastating wind hazards in East Asia." *Natural hazards* 92.3 (2018): 1321-1353
14. Steffen, Fredrik. "Wind-induced vibrations in high-rise buildings." *TVSM-5000* (2016).

15. Krishna, Prem, Krishen Kumar, and N. M. Bhandari. "IS: 875 (Part3): Wind loads on buildings and structures-proposed draft & commentary." *Document No.: IITK-GSDMA-Wind* (2002): 02-V5.
16. https://www.engineeringtoolbox.com/young-modulus-d_417.html
17. https://docs.blender.org/manual/en/latest/physics/rigid_body/properties.html
18. Canakci, Hanifi, et al. "Friction characteristics of organic soil with construction materials." *Soils and Foundations* 56.6 (2016): 965-972.
19. Fu, Feng. *Design and analysis of tall and complex structures*. Butterworth-Heinemann, 2018.
20. Lakshmanan, N., et al. "Basic wind speed map of India with long-term hourly wind data." *Current Science* (2009): 911-922.
21. Holmes, J.D., 2001. Wind Loading of Structures. Spon Press
22. https://pdfslide.net/documents/study-note-on-dynamics.html
23. https://blender.stackexchange.com/questions/199954/how-to-find-out-the-exact-frequency-of-f-curve-modifiers
24. https://docs.blender.org/manual/en/latest/animation/armatures/bones/properties/bendy_bones.html
25. https://sciencing.com/convert-wind-speed-force-5985528.html
26. https://learn.unity.com/tutorial/intro-to-the-unity-physics-engine-2019-3?uv=2019.4