iJournals: International Journal of Software & Hardware Research in Engineering (IJSHRE)
ISSN-2347-4890
Volume 10 Issue 1 January 2022

A Computational Approach to Learning Music

Siddh Merchant
Podar International School
siddh.merch@gmail.com

Reetu Jain
Chief Mentor, On My Own Technology
reetu.jain@onmyowntechnology.com

Abstract- Learning the guitar is not an easy task, several people have tried methods like online lessons and even physical ones, yet progress is slow. Due to the above, most quit whilst others find marginal levels of success. Research has been conducted to design a different approach to learning and practicing the guitar. Its purpose is to establish a system in which a user can create music without having to explore the intricacies of music theory, making the task easier and thus more achievable for the general population. By doing so, it strives to make music a more approachable domain for those who wish to explore it. In the process, I devised a unique way of approaching the instrument; one that avoids theory and draws focus on musical sense. Using open CV and python, I designed a program that takes scales as inputs and provides a harmonious sequence of notes as an output. Although based on the guitar, the theory explored in the project can be applied to any fretted instrument that holds a discrete set of notes. Using this program, general users can have access to instruments and consequently the benefits associated with playing music. This opens doors to music therapy and exercises used to focus/develop certain parts of the brain.

I. INTRODUCTION

Music is an integral part of our lifestyle. It has capabilities beyond the levels we initially tend to allocate. Although recreational, music tends to show several psychological benefits and thus, changes in behavior and brain structure. Psychological studies like one conducted by Samata R. Sharma, MD, MPH, and David Silbersweig, MD on neurobiological effects of music on the brain, suggest that music activates several parts of the brain and is capable of "altering both brain structure and function following early and repeated exposure". Furthermore, an investigation conducted by Roger E. Beaty on the neuroscience of improvisations suggests that "improvisation taps domain-general processes such as divergent thinking (Beaty et al., 2013) and cognitive flexibility (de Manzano and Ullén, 2012b)." By the above, studies conducted by Alessi shivonen, Teppo sarkamo, Mari tervaniemi, and Vera leo suggest that along with divergent thinking, users tend to show increased levels of focus and improved memory.

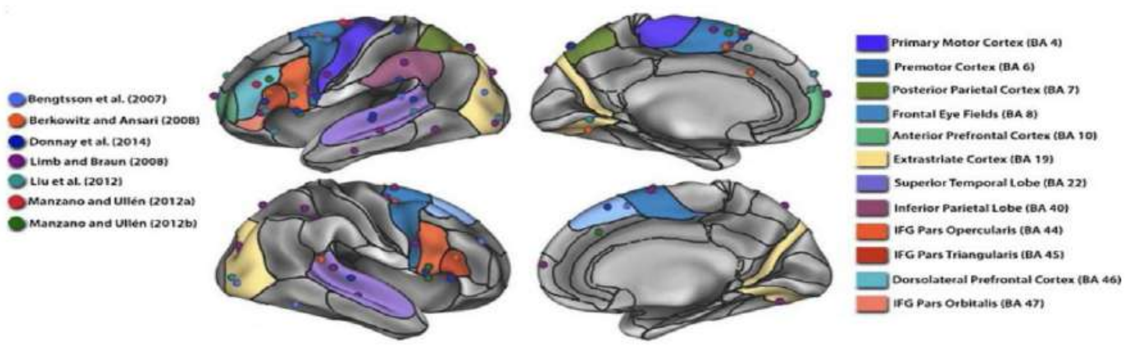

Bengtsson et al. (2007)
Berkowitz and Ansari (2008)
Donnay et al. (2014)
Limb and Braun (2008)
Liu et al. (2012)
Manzano and Ullén (2012a)
Manzano and Ullén (2012b)
Primary Motor Cortex (BA 4)
Premotor Cortex (BA 6)
Posterior Parietal Cortex (BA 7)
Frontal Eye Fields (BA 8)
Anterior Prefrontal Cortex (BA 10)
Extrastriate Cortex (BA 19)
Superior Temporal Lobe (BA 22)
Inferior Parietal Lobe (BA 40)
IFG Pars Opercularis (BA 44)
IFG Pars Triangularis (BA 45)
Dorsolateral Prefrontal Cortex (BA 46)
IFG Pars Orbitalis (BA 47)

Figure 1:"Visualization of activation foci reported in select studies of improvisation"

Detailed studies conducted by Roger E. Beaty show the activation of the above foci during musical improvisation. Furthermore, observations suggest stimulation in the left inferior frontal gyrus, anterior cingulate cortex, and the medial prefrontal cortex, involved in comprehension, error management, and mood control respectively. Rhythmic improvisation stresses the above whilst also activating regions in the left superior and inferior parietal lobules associated with emotion and interpretation of senses.

From a psychological standpoint, an article written by Heather Craig, BPsySc, confirms that music therapy and consequently, the playing of an instrument, tends to reduce anxiety and stress in a patient and consequently improves healing, while research conducted on the effectiveness of music therapy by Hiroharu Kamioka and Kiichiro Tsutani among others suggests improvement in depressive symptoms, schizophrenia, sleep quality and Parkinson's disease.

For the above reasons, playing and listening to music carries immeasurable benefits; benefits that most miss out on because of lack of interest, reduced self-confidence, the complexity of music theory, and lack of success or development in their musical paths as suggested by an article written by Esther murimi, a teacher at Merriam school of music and a statistical study conducted by ABRSM in the UK.

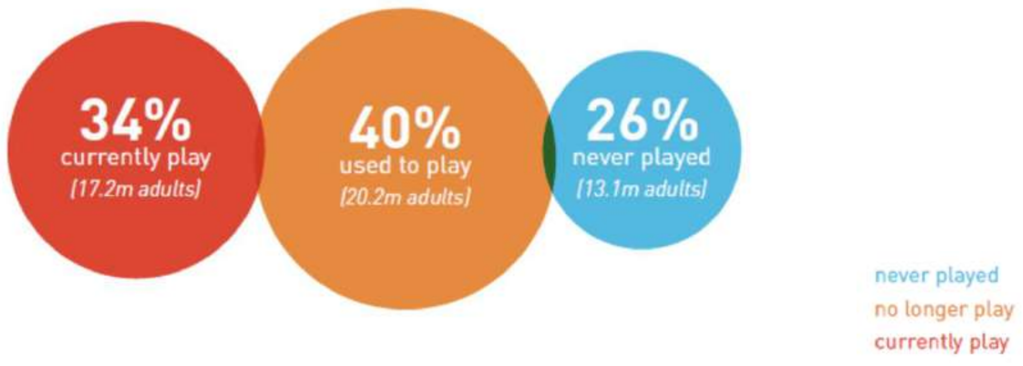


Figure 2: "Pictorial representation of percentage of adults that play instrumental music according to a study conducted by ABRSM in the UK"

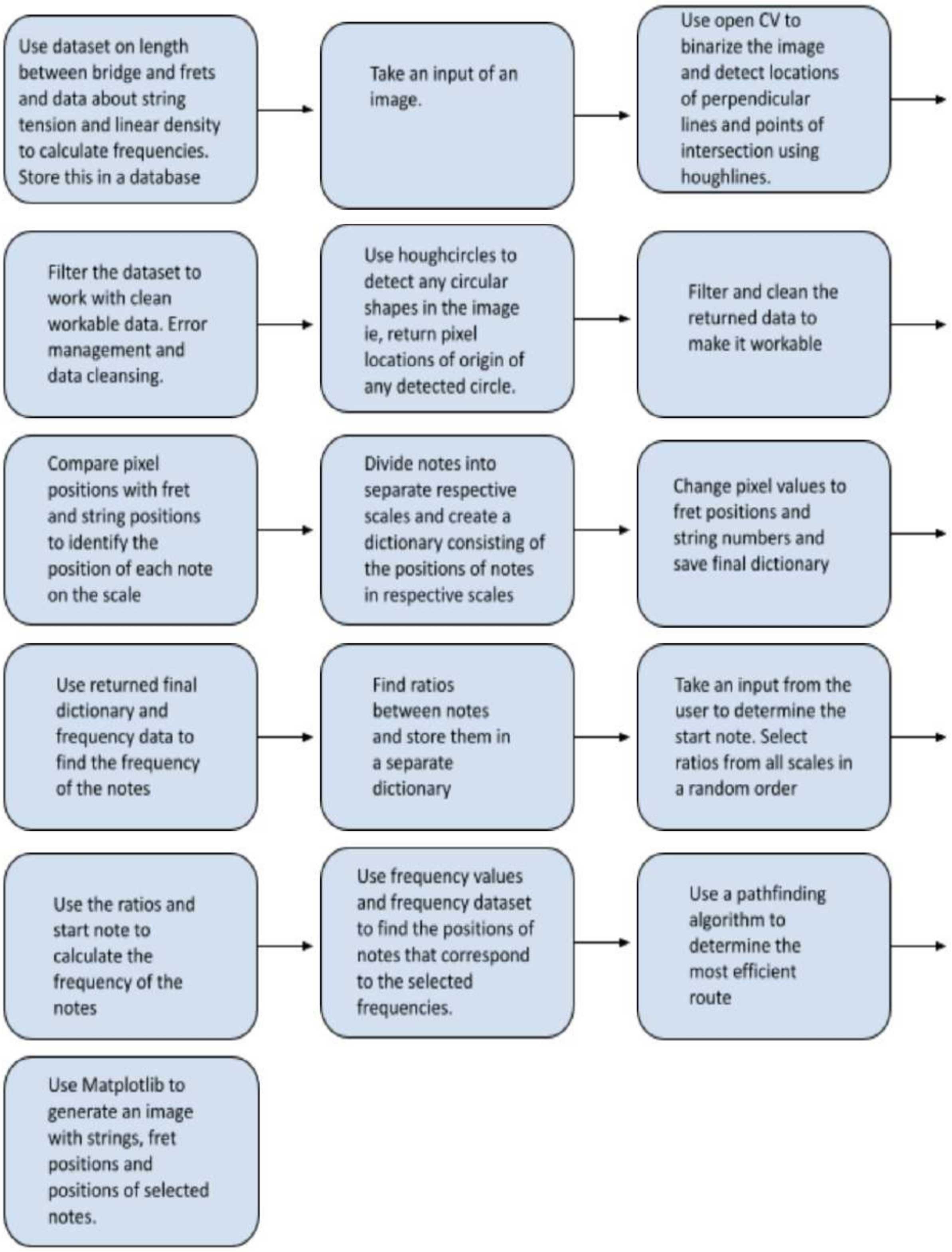
As suggested by the above graph, the percentage of students/adults who stop playing instruments is the greatest. Loss of interest is a major factor due to advancing levels of theory and decreased pace of progress/improvement. The purpose of this program is to offer a separate path, avoiding the conventional theoretical approach. It avoids musical knowledge but rather focuses on musical sense, allowing a user to express harmonies over a discrete set of harmonic notes; allowing timing, progression, and pattern of selection to be the variables controlled by the user. The aim of this is to reduce the complexity associated with playing an instrument and allow users to access its psychological benefits, nonetheless.

II. LITERATURE REVIEW

The project is based on three programs. One maps out the frequency of every possible note that can be played on the guitar. The other performs image analysis on imputed scale images using open CV and returns the frequency of detected notes on a scale. The final program uses these scale positions and ratios between notes to create a sequence of harmonic notes. It takes in a user input to determine the position of the start note and uses a pathfinding algorithm to construct an efficient route for the user to follow on the guitar. Note positions are defined as coordinates in the x and y plane where the y axis determines the string number and the x-axis determines the fret position. During the construction of the final image, a visual depiction of the guitar neck, the strings, the frets, and the locations of the notes that have been selected by the program is formed.

III. METHODOLOGY

Algorithm Flowchart:


Use dataset on length between bridge and frets and data about string tension and linear density to calculate frequencies. Store this in a database
Take an input of an image.
Use open CV to binarize the image and detect locations of perpendicular lines and points of intersection using houghlines.
Filter the dataset to work with clean workable data. Error management and data cleansing.
Use houghcircles to detect any circular shapes in the image ie, return pixel locations of origin of any detected circle.
Filter and clean the returned data to make it workable
Compare pixel positions with fret and string positions to identify the position of each note on the scale
Divide notes into separate respective scales and create a dictionary consisting of the positions of notes in respective scales
Change pixel values to fret positions and string numbers and save final dictionary
Use returned final dictionary and frequency data to find the frequency of the notes
Find ratios between notes and store them in a separate dictionary
Take an input from the user to determine the start note. Select ratios from all scales in a random order
Use the ratios and start note to calculate the frequency of the notes
Use frequency values and frequency dataset to find the positions of notes that correspond to the selected frequencies.
Use a pathfinding algorithm to determine the most efficient route
Use Matplotlib to generate an image with strings, fret positions and positions of selected notes.

Concepts and formulas used:

The concepts explored by the project revolve primarily around coordinate geometry and wave theory. Stationary waves play a huge role in determining the frequency of each note and coordinate geometry is used extensively in the generation of the final output image as well as detecting intersection points of lines in the input image.

Stationary waves:

Stationary waves or standing waves are defined as waves produced because of superposition between two progressive waves traveling in opposite directions, each having the same amplitude and frequency, with the same speed. For our purposes, we will be exploring those stationary waves that exist between two nodes. In the guitar, the two nodes exist at the bridge and the nut. When we play a note, we change the position of one node and consequently the distance between the two nodes. For any stationary wave of length L, the wavelength of produced sound is defined by the following formula:

Lambda = 2L/n where: lambda = wavelength, L = distance between nodes, and n = the harmonic of the stationary wave. (For our purposes, we always use the first harmonic hence, n = 1). It is important to recognize that for a string, the frequency of a stationary wave is dependent on:

- Length between two Nodes
- Tension of the string
- Linear density of the string
- Harmonic of the stationary wave.

Off the above, we will be keeping one dependent variable, one independent variable and the rest would be defined as constants. To this project, the length between nodes is the independent variable and frequency is the dependent variable. The respective tensions and linear densities of the strings are kept constant at standard tuning.We can use the formula for the wavelength of stationary waves and the universal wave equation to find the frequency of the produced wave.

$$V = f_x \times \lambda$$

$$V = f_x 2\frac{L}{n}$$

$$\frac{(V \times n)}{2L} = f$$

Note that we do not know V yet. The velocity of sound in a string is dependent on the root of the ratio between the tension and linear density of the string hence, V is written as follows:

$$V = \sqrt[2]{\frac{T}{\mu}}$$

where T = tension and u = Linear density.

Now that we know V, we can plug this into the previous equation for frequency and obtain a formula in terms of all the variables we can control.

$$(V \times n)/2L = f$$

$$(T/u)^{1/2} \times n/2L = f$$

$$((T)^{\frac{1}{2}} \times n)/(\left(u\right)^{\frac{1}{2}} \times 2L\right) = f$$

where T = Tension, u = Linear density, n = Harmonic, L = Length between nodes,
As we keep the tension, linear density, and harmonic of wave constant for each respective string, we can theorise that increasing the length between nodes, decreases the frequency of sound and vice versa.

Graph of f vs $\frac{1}{L}$:

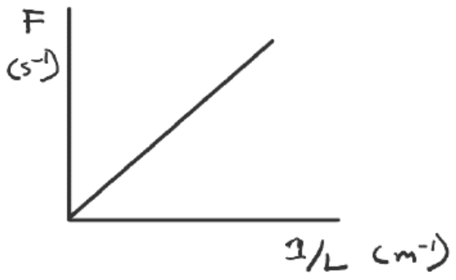


Figure 3: "The graph for frequency versus inverse of length"

Where the gradient is: $m = (n \times (T)^{\frac{1}{2}})/(2 \times (u)^{\frac{1}{2}})$

If the graph passes through the origin, the relationship is satisfied.Dimensional analysis of the formula:

The objective of the analysis is to equate RHS to LHS and verify whether the equation used is correct or not. If RHS equals LHS after dimensional analysis, the formula is correct and vice versa. Perform dimensional analysis, we must convert all derived quantities and express them as base quantities. It is important to recognise that n and two are both constants in the formula and have no units.Frequency is defined as the number of waves passing a point in one second. It can be expressed in terms of time in the following manner:

$F = {}^{1}/_{t}$where it is the time of the wave. The unit of frequency is hertz = Hz which can also be expressed as s^-1 where s = second, a unit of time.Our formula, $F = \frac{n \times (T)^{\frac{1}{2}}}{(2L \times (u)^{\frac{1}{2}})}$consists of derived quantities hence, we must express each in terms of base quantities first.

$$T = Mg = kg \times ms^2$$

$$u = M/L = kg\, m^1$$

$$F = \frac{(T)^{\frac{1}{2}}}{L \times (u)^{1/2}}$$

$$F = \frac{(kg \times m \times s^2)^{\frac{1}{2}}}{m \times (kg\, m^{-1})^{1/2}}$$

$$F = kg^{1/2} \times m^{1/2} \times s^{-1} / \, m \times kg^{1/2} \times m^{1/2}$$

The ‘kg^½’ term is present in both the numerator and the denominator hence can be replaced by 1.

$$F = 1 \times m^{1/2} \times s^{-1}/m^{1/2}$$

The ‘m^½’ term is present in both the numerator and the denominator and can be replaced by 1.

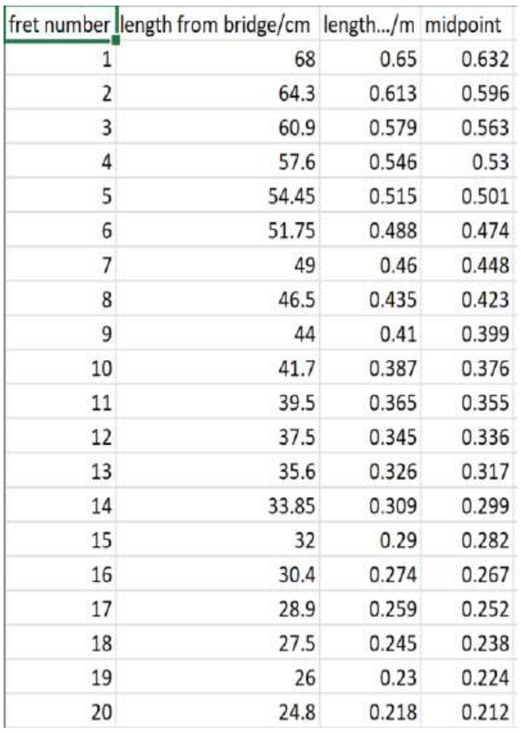
$$F = 1 \times 1 \times s^{-1}$$

$$F = s^{-1}$$

We derived earlier that $F = s^{-1}$hence, $s^{-1} = s^{-1}$therefore, RHS = LHS and the formula is proven to be correct.

IV. ALGORITHM

The first step is to create a database that consists of all the frequencies that can be played on the guitar. As it is a fretted instrument, there is a discrete set of notes that can be played. As mentioned above, the frequency changes with distance hence, the closer the frets are to the bridge, the higher the pitch of sound. Calculate the frequency, we need a database of length between the midpoint of frets and bridge of the guitar.



fret number	length from bridge/cm	length.../m	midpoint
1	68	0.65	0.632
2	64.3	0.613	0.596
3	60.9	0.579	0.563
4	57.6	0.546	0.53
5	54.45	0.515	0.501
6	51.75	0.488	0.474
7	49	0.46	0.448
8	46.5	0.435	0.423
9	44	0.41	0.399
10	41.7	0.387	0.376
11	39.5	0.365	0.355
12	37.5	0.345	0.336
13	35.6	0.326	0.317
14	33.85	0.309	0.299
15	32	0.29	0.282
16	30.4	0.274	0.267
17	28.9	0.259	0.252
18	27.5	0.245	0.238
19	26	0.23	0.224
20	24.8	0.218	0.212

Figure 4: "Database for fret number and length from bridge position of the guitar to the midpoint of each fret"

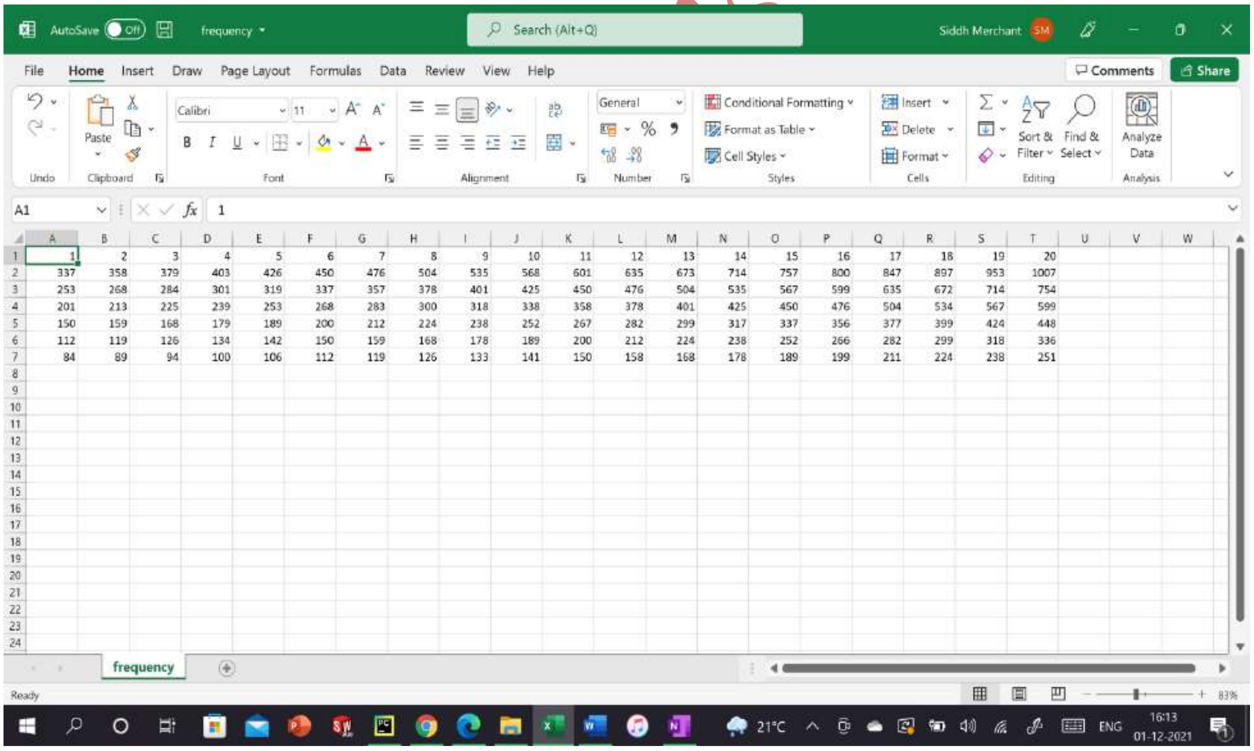
The equation also deals with the linear density and tension of each string at standard tuning. Although these are kept constant, we need the value of the constants to calculate the frequency. The following database is used for the above:

Fender Super 250L's

String	Hi E (~.009)	B (~.011)	G (~.016)	D (~.024)	A (~.032)	Low E (~.042)
MASS (Kg)	3.32E-04	4.88E-04	1.07E-03	1.92E-03	3.18E-03	5.42E-03
DENSITY(Kg/m)	3.09E-04	4.77E-04	1.03E-03	1.62E-03	3.50E-03	5.78E-03
DENSITY(Kg/Vol)	7.97E+03	7.83E+03	7.99E+03	5.48E+03	6.75E+03	6.60E+03
FREQUENCY(Hz)	329.63	246.94	196.00	146.83	110.00	82.41
TENSION (N)	56.40	48.86	66.61	58.55	71.11	65.90

Figure 5: "Database for linear density, frequency, tension, density and mass for each string at standard tuning. Note: this dataset is applicable to fender super 250L's. The database for strings from different manufacturers are likely to have different values"

(Keep in mind that the linear density of a string is a material specific property hence, it will change relative to the type of guitar strings used. For our purposes, we use fender 250L's.)Using these values and our database on the distance between frets and bridge of the guitar, we can move on to calculating the frequency of every note on the guitar.Calculate the above, the program Frequency.py is used. It accesses the previous two datasets using pandas and appends the values into a list. The data for linear density and tension is sorted into a dictionary relative to the corresponding string. Using the above and the list for distance, the program executes the following function. The calculated frequency for each string is stored in a separate list, each of which is appended into a dictionary relative to the string. Using this dictionary as a dataset, we can create another csv file holding the frequency of every note on the guitar.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	337	358	379	403	426	450	476	504	535	568	601	635	673	714	757	800	847	897	953	1007
3	253	268	284	301	319	337	357	378	401	425	450	476	504	535	567	599	635	672	714	754
4	201	213	225	239	253	268	283	300	318	338	358	378	401	425	450	476	504	534	567	599
5	150	159	168	179	189	200	212	224	238	252	267	282	299	317	337	356	377	399	424	448
6	112	119	126	134	142	150	159	168	178	189	200	212	224	238	252	266	282	299	318	336
7	84	89	94	100	106	112	119	126	133	141	150	158	168	178	189	199	211	224	238	251

Figure 6: "Database for the frequency of every note on the guitar from frets 1 to 20 for all strings"

Image analysis:

For image analysis, we use OpenCV. Using this library we perform a series of functions, namely, stretching and morphing (resizing), binarizing, contour detection, and shape detection. Furthermore, we also use _collections, NumPy, math, matplotlib, pandas and random to perform other functions in the program.

We perform these functions on the following image (A minor scale):

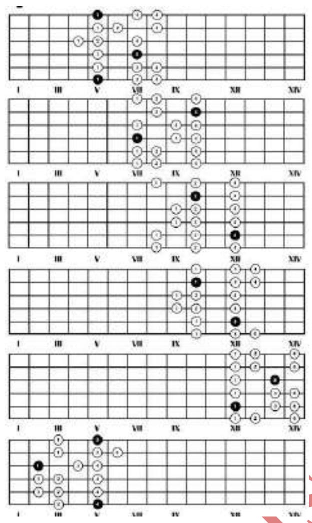


Figure 7: "Image used in the process"

To Execute the above task, we define a class image analysis and add a constructor method along with two other methods: calibrate and read. Within the constructor we instantiate the name of the file as in, the image we want to use, and hardcode certain parameter values for the Hough circles algorithm for simplicity along with the required height and width for the resizing of the image.We add code to calibrate the image to the required specifications. Do this, we use the resize function from the OpenCV library.The resize function consists of five important parameters, the image name or src, size or desired size of the output image, fx to scale the horizontal axis, fy to scale the vertical axis, and interpolation to estimate values at unknown points. As we are simply resizing the image, the only parameter we are concerned with is size to which we assign the values hardcoded in the constructor.Post correction, the image is exposed to a series of functions. After being read, its colour is converted from RGB to grayscale using cv2.cvtColor. After conversion, we perform canny edge detection where the first parameter takes in the grayscale image and the other two provide the lower and upper quartile for the range of intensities of gradients drawn on shapes that can be classified as edges. After doing so, we binarize the image using the threshold method and then crop it to obtain only the first scale on the image. This is important as it allows us to determine the dimensions, we wish to offset the coordinates obtained post Hough line transform to obtain a pair of coordinates on a line.

The Hough line transform algorithm is executed by a method named Hough lines which is part of the open cv library. It returns a 2d array of all the lines detected. Detect lines, Hough Lines plots the polar equation of a line on the cartesian plane to form curves. The point of intersection of these curves and the number of curves passing through said point will judge whether those rho and theta values translate to a line or not. In our program, we have applied the threshold to be 200: 200 curves must pass through the same point on the cartesian plane to consider the rho and theta values of the point of intersection as a line.

After finding the various lines and storing them in a 2d array, each value is translated into cartesian coordinates and offset to the edges of the image. These values are then fed into the cv2.line function as parameters and a line is plotted on the image. The following image is the output of the above process:

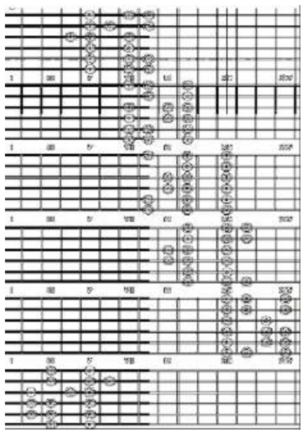


Figure 8: "Picture of the binarized image"

After plotting the lines, we must determine the points of intersection between the horizontal and vertical lines. Each of these lines are defined as vectors in terms of rho and theta hence we can use linear algebra to solve for the point of intersection. We use np.Linalg.solve to do so. It solves a system of matrix equations in the form Ax = b where x is the vector for which we are solving. In our scenario, np.cos theta represents changes to I hat or the unit vector in the x direction and np.sin theta represents changes to j hat or the unit vector in the y direction. This change is applicable for coordinates where the result equals the value of rho for the respective lines. For those coordinates where this change is applicable, the function returns the x and y value of the coordinate as a 2d list.

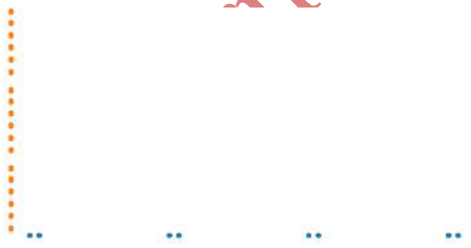


Figure 9: "A zoomed in section showing the presence of noise in the data and thus the need for a data cleansing algorithm"

The problem we face after this step is noise. Due to double line detection, there are several redundant coordinates surrounding the same point hence, they need to be dealt with. The method average_out () deals with this. It takes in a sorted list of integers as a parameter and traverses both ends of the list from the midpoint. After closely examining the pixel values for the selected coordinates, the error bar was estimated to be between 1 and 5 pixels. The midpoint value is compared with (we subtract the two to find the difference between pixel values) the previous value in the list in the first half and compared with the next value in the list in the second half consequently, its value is decremented and incremented, respectively. Any value within the range 1 to 5 will be considered as a repeat and the original value will be appended into a list called repeats. After sorting this list, we compare this dataset against the original to remove any repeated values from the original array. There is however, one major red flag that still must be dealt with. There is a chance that Hough Lines may accidentally detect an extra line in the image. These extra lines go amiss in the above procedure and must be dealt with separately. Do so, we perform a similar function as the above; however, this involves one additional step. After calculating the differences between pixel values of selected coordinates and storing them in a separate array, we perform another step to find the differences between these values. This helps us detect the anomaly in our result as for most values, the difference between them must be constant; however, the presence of an extra coordinate would disturb this sequence and result in an anomaly which can be picked by another error bar: this time defined as -5 to +5. Any values that fall under this criterion are appended into another list defined as in-between and consequently removed from the main array. The function returns the array as a cleaned list.

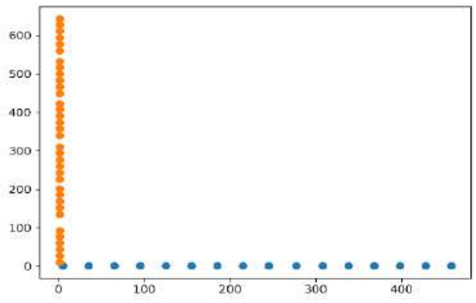


Figure 10: "A scatter plot showing the locations of strings and frets in our imputed image. This is post data cleansing"

The above image is a representation of all the detected points. The points in orange represent the positions of strings on each scale whereas the points in blue represent the fret positions.

The purpose of cleaning the data is to make it more comprehensible and usable. By cleaning the dataset, we can determine the string and fret position of notes on the scales; however, to do so, we must first determine the location of the origin of the notes on the scales. Do so, we use the method Hough circles. Unfortunately for us, the parameters of Hough circles tend to be very image centric hence, parameter values that might work for one image may not work for another. For this reason, these values have presently been hardcoded. Hough circles determine the origin of any circle by drawing circles on every point on the circumference of a detected circle. Every drawn circle may or may not intersect at a fixed point. If the number of circles passing through a fixed point is greater than our accumulator threshold value, we can define that point as the origin of a circle. This value is referenced here as param2.

The radius of the circles we wish to detect lie between 5 and 8 for the image we have chosen, and each circle can be a minimum of five pixels away from the other. All these properties are image centric and are likely to change if we use a different image.

However, performing this algorithm on our selected image yields the following result:

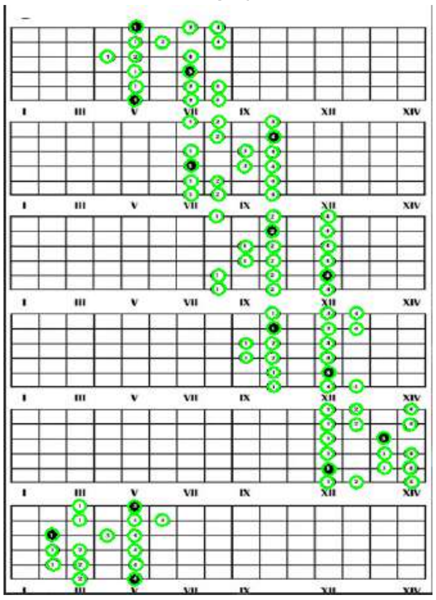


Figure 11: "Image showing the detected circles after Hough circles is carried out"

Hough Circles returns a 3d array of the x and y coordinates for the origin of each detected circle. These coordinates must now be compared against the string values and fret positions obtained earlier to determine where the note lies on the fretboard. The program to do this is extensive. The first step is to translate the y coordinate of the origin of the circles in terms of the y coordinates of the lines obtained in the previous algorithm. It is important to recognize that although the circles lie on the line, it is highly unlikely that their y coordinates exactly match. However, it can be inferred that the difference between the two is likely to be small hence, we can introduce an error bar for the values from -5 to 5 pixels. Any y coordinate of the origin of a circle within range will be corrected to the y coordinate of the string we are comparing it to.

For the x coordinate, we must compare the x coordinate of the origin of the circle to the fret positions determined in the earlier algorithm. Keep in mind that each note does not exist on a fret but is present between frets hence, we must create a 2d array of the range within which a particular note can lie after which, we simply iterate through the list of x coordinates of the origin of the circles and compare them to the range. Values that fit within the range will in turn be replaced by the range in the original list. Of course, we cannot use a range to define the position of a note hence, we must replace this value by fret number in the original list. This can be done by using a sorted 2d list of the range and replacing each value with the index position of the array in the sorted list. It is important to recognise that not all scales in the image begin from the first fret (although this is the case here) hence, we must correct the x coordinate by offsetting it by the required amount. This unfortunately cannot be automated easily thus, for simplicity, the function correct_for_x simply requests a user input to determine how much each x coordinate must be offset by depending on what scale the note is on. Our note can thus be defined by the following statement: fret number, y coordinate of the string. The next step is to correct the y coordinate to string number.

The first step in the above process would be to divide our dataset into the scales they represent. We can do this by comparing the y coordinate values of each note. Notice that in the scatterplot, the orange dots are localized in specific spots in groups of six. This is a good thing as there are six strings on a guitar. Another distinguishing factor is that each group is separated by a larger distance than those in the group. We can thus compare (find the difference between) the y coordinate of the pixel values. Within a group this difference would be constant but as soon as a change in group is present, this difference would change and would be large. We can determine this by assigning the following range: values within -20 and twenty pixels of each other are within a group whilst values out of this range would be considered as the start of a new group. Consequently, the dataset is split into separate groups representing the notes for separate scales. This data can be stored in a dictionary.

After the organization is complete, we can define a method swap. Swap instantiates a list containing string numbers from 0 to 5 and takes in a 2d array as a parameter and stores the y coordinate of the pixels representing the notes into a separate list. This list is converted into a set to get rid of repeating values and then converted back into a sorted list. The two corresponding values of string number and y coordinate of the pixels are appended into a 2d list which can then be used to replace the respective y coordinates with string numbers in the original array. The array is then returned.

We can access each array from the dictionary using dictionary keys, passing each array into the function swap, and replacing the original array with the returned array in our final dictionary. We now have a dictionary that is defined in terms of fret position and string number. It can thus be passed onto the next step: Finding the frequency and generating an output image.

Finding the frequency:

Class Find Frequency inherits from class Image Analysis hence, it has access to its constructor and methods. Find Frequency has two methods where one is the constructor and the other is calc_ratios. The objective of this program is to use the dictionary returned post image analysis to find the frequency of each note on the scale. We can then use the ratios between these notes to construct our own pattern on the fretboard.

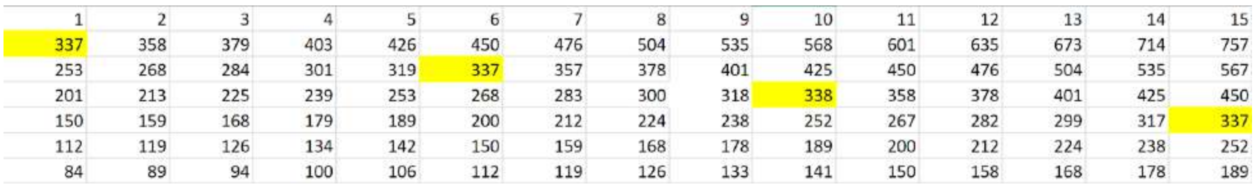
Ratios are key for this algorithm to work. Each harmonious note following the first can be modelled by the following equation 2^n/12 where n is the number of semitones. The scales we use oscillate between the first and second semi tones giving us ratios equal to 1.06 and 1.12, respectively. But why is this the case? Such ratios typically cause a pattern to emerge between the rhythm of two continuous notes. A sound wave is quite simply beats played in rapid succession and when two harmonious notes are played together, if we slow them down, a pattern emerges in their rhythm. Our brains tend to pick this pattern up and consequently we find it harmonious. The simpler the pattern, the more harmonious the notes sound together and the more complex, the more cacophonous. This is exactly how scales are formed.

The first step is to read our frequency.csv file and access the frequency values. We can do this using panda. By using the pd.read_csv() function, we can store the above data in the form of a pandas dataframe. As we have the positions of the frequency of the note already in terms of fret position and string number, all we need to do is use those coordinates to access the frequency value corresponding to that position and append this value in place of the coordinates. We must repeat this process for all the scales in the dictionary.

We can now calculate the frequency of each note using the formula tn + 1/ tn where it is the term and n is the index position of the note in the list. We replace each list containing the frequency values of the scales with the list of ratios. Selecting a random number of ratios from each scale, we can construct a different pattern for every try. Using the inbuilt library random, we select 1-5 values from a certain scale before moving onto another. We can ensure that these ratios sound harmonious together by offsetting the start index for the next list by the amount we traverse on the previous list. We append all selected ratios into a new 1d array called ratio_list.

Using the selected order of ratios, we can calculate the frequency of the notes that make up our lick. The user, however, must input the first notes position on the fretboard. Using this note as the first value of our sequence, we can construct a list of frequencies corresponding to the ratios we have selected. The first note will be multiplied with the first ratio in our list to obtain the second note in our progression. The second will be multiplied with the second ratio in the list and so on. Each calculated frequency is then appended into a separate list known as frequency list.

The next step is to find where these frequencies lie on the fretboard. It is important to recognize that the same note can be played at several positions on the fretboard. For instance, the note corresponding to the frequency 337 +- 1Hz can be played in the following positions:



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
337	358	379	403	426	450	476	504	535	568	601	635	673	714	757
253	268	284	301	319	337	357	378	401	425	450	476	504	535	567
201	213	225	239	253	268	283	300	318	338	358	378	401	425	450
150	159	168	179	189	200	212	224	238	252	267	282	299	317	337
112	119	126	134	142	150	159	168	178	189	200	212	224	238	252
84	89	94	100	106	112	119	126	133	141	150	158	168	178	189

Figure 12: "Image showing the various positions the same note can be played on the fretboard"

This makes construction of the lick slightly complicated. The first step however is to detect the multiple positions in which a single note can be played. A dictionary can be constructed for this process where each frequency can function as a key corresponding to a 2d list of the positions where the note can be played on the fretboard. It is important to recognize that the calculated frequencies are not likely to be equal to the frequencies in the database and may differ by lesser amounts hence a small error bar must be included (-3 to 3 Hz in this program) to detect all positions. Using this and a pathfinding algorithm, we can construct an efficient and viable route for the guitarist to follow on the fretboard.

find_closest () is the function we use to select the positions. We know our start node as that has been imputed by the user. Relative to this, the second note is detected. Keep in mind that there might be one or more positions at which the note can be played. We can use the distance formula of coordinate geometry to calculate the length between the initial note and the new note for each respective position. We can append these values into a new list known as list_of_distance whilst the corresponding coordinates can be stored in a separate list. Storing these values in ascending order, we can find the index position of the minimum distance and second minimum distance in the list. The coordinates of the point used to calculate each distance value, can be retrieved by applying these index positions to the list of coordinates. The reason we do the same process for the second minimum distance is to ensure we have a backup value. We need this because if we were to select the minimum distance at all points, it is highly likely that we construct a pattern such that all the notes are played on the same string. This is not helpful as it would mean that accessing all the notes in the lick would become extremely hard. An efficient route would be one that is concise on the fretboard utilising as many strings as possible. For that reason, we place a limit on the total number of notes that can be played on the same string. We switch to the second minimum distance if the minimum distance corresponds to a note on the same string as the previous note and the count of notes on the string has already reached its maximum value. We append all the selected coordinates into another list known as final and return this to the main program.

The last step of our program is to construct an image. As we have the coordinates, we can simply use the scatter plot function of matplotlib to display those points. However, to make it more understandable, the construction of strings and frets are required. We can plot them using the hlines and vlines function of matplotlib, constructing six strings, represented in green, and 20 frets, represented in blue. The following image is our final output:

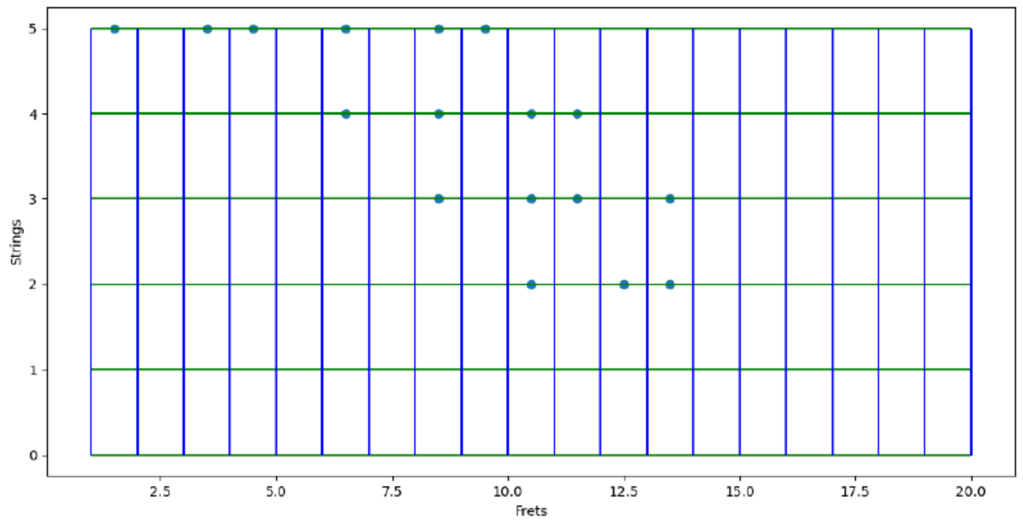


Figure 13: "Final output image of the program"

For the above, the guitarist can follow the notes from left to right where the topmost string represents the low E and the bottom most string represents the high E.

RESULT & CONCULSION

The above data highlights a viable model for enthusiasts wishing to pursue the guitar and experts who wish to gain a deeper understanding of the fretboard. Through the theory explored in the methodology, it can be assured that the notes would sound harmonious when always placed together as the ratios used during this procedure translate directly to the equation used to calculate ratios based on semitones.

It is important to recognise that the use of this program is focused on two groups: People suffering from psychological or physiological disorders that can be resolved through music therapy and intermediate to high level music players that have a general understanding of the instrument. The output image is one that helps assist in note selection only. While it is true that anyone can pick up a guitar and play these notes, it would not be recommended for any beginner to directly start off like this as they are not likely to have any prerequisite knowledge of what they are doing. Consequently, the effects of blindly following this image would oppose its intention.

The purpose is to inspire creativity in a subject. Notes are a big gap that block a subject from the medicinal benefits of playing music and thus, they must be overcome to access them. By removing this impedance, we provide a greater opportunity to express creativity and more importantly, allow access to these benefits. For those suffering from any illness, learning a new instrument from scratch is a daunting task; however, this can be made several times easier through the above procedure. Following automated paths is not analogous to learning the theory, practicing scales, and mastering the instrument. It does, although carry similar effects. It provides ground for improvisation and hence, space for creative expressionism. Medicinal benefits are primarily associated with this quality. As this program acts as a stimulant of the above, it has the potential to improve the rate of success and completion of music therapy, inturn providing those suffering from disease a greater chance of recovery.

Future scope:

At this stage, the program is merely a working solution and is not modelled to serve the consumer. The purpose of this program is to create a wider platform for music lovers and aspirators to grow on. Do so, it must be able to manage different types of image formats and consequently varying inputs. It should also be fast, producing a much more intuitive and understandable output so that both an expert and layman can read and understand it. It is important to recognize that this program is not designed to replace music theory entirely hence, it should also display the names of the notes it selects.

There is a lot of space for AI in this algorithm, especially programs that deal with deep learning. Introducing a convolutional neural network would allow greater flexibility especially using hyperparameters. When it comes to dealing with images, as discussed earlier, conducting programs like Hough Circles and Hough Lines involve specific parameter settings to allow proper detection. This leads to several issues when trying to automate the program; however, CNN can be used to solve this. By establishing a database of several types of different images and forms scales can be imputed in and storing their parameter values for Hough circles and Hough lines, we can compare each imputed image to understand which parameters best fit it. This has the potential to make the program less prone to errors and more flexible. Another important development would be working on program efficiency. At this stage, the program may consist of redundant or even inefficient approaches to solving certain problems. We can employ programming methods like recursion to increase efficiency and thus, decrease processing time and processor stress. Furthermore, during pathfinding more general algorithms like Dijkstra or A* could be used. Taking into consideration our specific requirements, we can modify these already existing efficient algorithms to further decrease processing time.

As outlined earlier, the medicinal benefits of playing music must not be wasted away. The psychological and physiological effects associated with the act has the potential to develop our cognitive capabilities hence, establishing an easier platform of access is essential. Considering the scale of human development in the past decades and extrapolating this statistic to the future highlights a greater need for creative minds in all fields. Introducing music at an early stage for the future generations can help achieve this. This platform simply provides an alternate and more direct path to accessing these benefits.

REFERENCES

1. 1md, Organization. “What Happens in the Brain When You Have Depression? Causes & Treatment.” *1MD*, 1MD, 5 Feb. 2015, https://1md.org/article/depression-brain-causes-treatment.
2. A;, Holm-Hadulla RM;Bertolino. “Creativity, Alcohol and Drug Abuse: The Pop Icon Jim Morrison.” *Psychopathology*, U.S. National Library of Medicine, https://pubmed.ncbi.nlm.nih.gov/24051493/.
3. Aalbers S;Fusar-Poli L;Freeman RE;Spreen M;Ket JC;Vink AC;Maratos A;Crawford M;Chen XJ;Gold C; “Music Therapy for Depression.” *The Cochrane Database of Systematic Reviews*, U.S. National Library of Medicine, https://pubmed.ncbi.nlm.nih.gov/29144545/.
4. Achilles, daryl. “Tensions of Guitar Strings - Course Websites.” *Tensions of Guitar Strings*, https://courses.physics.illinois.edu/phys406/sp2017/Student_Projects/Fall00/DAchilles/Guitar_String_Tension_Experiment.pdf.
5. Bapat, Krutika. “Hough Transform Using Opencv.” *LearnOpenCV*, 4 May 2021, https://learnopencv.com/hough-transform-with-opencv-c-python/.
6. Baycrest, Center for geriatric care. “Uncovering Why Playing a Musical Instrument Can Protect Brain Health.” *ScienceDaily*, ScienceDaily, 1 June 2017, https://www.sciencedaily.com/releases/2017/06/170601192721.htm.
7. Beaty, Roger E. “The Neuroscience of Musical Improvisation.” *Neuroscience and Biobehavioral Reviews*, U.S. National Library of Medicine, 16 Jan. 2015, https://pubmed.ncbi.nlm.nih.gov/25601088/.
8. GeeksforGeeks, organization. “Image Resizing Using Opencv: Python.” *GeeksforGeeks*, 15 Mar. 2021, https://www.geeksforgeeks.org/image-resizing-using-opencv-python/.
9. GeeksforGeeks, Organization. “Line Detection in Python with Opencv: Houghline Method.” *GeeksforGeeks*, 14 Feb. 2018, https://www.geeksforgeeks.org/line-detection-python-opencv-houghline-method/.
10. Iszáj, Fruzsina, et al. “Creativity and Psychoactive Substance Use: A Systematic Review.” *International Journal of Mental Health and Addiction*, Springer US, 20 Oct. 2016, https://link.springer.com/article/10.1007/s11469-016-9709-8.
11. Kamioka H;Tsutani K;Yamada M;Park H;Okuizumi H;Tsuruoka K;Honda T;Okada S;Park SJ;Kitayuguchi J;Abe T;Handa S;Oshio T;Mutoh Y; “Effectiveness of Music Therapy: A Summary of Systematic Reviews Based on Randomized Controlled Trials of Music Interventions.” *Patient Preference and Adherence*, U.S. National Library of Medicine, https://pubmed.ncbi.nlm.nih.gov/24876768/.
12. Kim, Jae-Hun, et al. “Defining Functional SMA and Pre-SMA Subregions in Human MFC Using Resting State Fmri: Functional Connectivity-Based Parcellation Method.” *NeuroImage*, U.S. National Library of Medicine, 1 Feb. 2010, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2819173/.
13. Körting, Thales Sehn, director. *How Circle Hough Transform Works*, 26 Feb. 2020, https://www.youtube.com/channel/UCSd_7rz5nzSnzUYbjaCXC5g.
14. Lathiya, Ankit. “Numpy Linalg Solve() Function in Python Example.” *AppDividend*, 9 Nov. 2020, https://appdividend.com/2020/11/09/numpy-linalg-solve-function-in-python-example/.
15. Life2CodingTechnology Related Blog at Life2CodingFeel free to contact us for your any kind of technical problems. We are here to help you., organization. “How to Resize Images in Opencv Python Using Different Interpolation Methods.” *Life2Coding*, 6 Nov. 2021, https://www.life2coding.com/how-to-resize-images-in-opencv-python-using-different-interpolation-methods/.
16. “Music and the Brain.” *Neurobiology*, Harvard Medical School, https://neuro.hms.harvard.edu/centers-and-initiatives/harvard-mahoney-neuroscience-institute/about-hmni/archive-brain-1.
17. Numpy, Organization. “Numpy.linalg.solve¶.” *Numpy.linalg.solve - NumPy v1.21 Manual*, https://numpy.org/doc/stable/reference/generated/numpy.linalg.solve.html.
18. OpenCV, Organization. “Feature Detection.” *OpenCV*, https://docs.opencv.org/4.x/dd/d1a/group__imgproc__feature.html.

19. OpenCV, organization. “Geometric Image Transformations.” *OpenCV*, https://docs.opencv.org/4.x/da/d54/group__imgproc__transform.html.
20. OpenCV, Organization. “Hough Circle Transform.” *OpenCV*, https://docs.opencv.org/4.x/d4/d70/tutorial_hough_circle.html.
21. OpenCV, organization. “Hough Line Transform.” *OpenCV*, https://docs.opencv.org/4.x/d9/db0/tutorial_hough_lines.html.
22. “Opencv-Python图像处理4.” *Opencv-python图像处理4_热衷开源的Boy-CSDN博客*, https://blog.csdn.net/weixin_39025679/article/details/104607033.
23. R, Nick, et al. “Detecting Circles in Images Using Opencv and Hough Circles.” *PyImageSearch*, 17 Apr. 2021, https://www.pyimagesearch.com/2014/07/21/detecting-circles-images-using-opencv-hough-circles/.
24. Raglio, Alfredo, et al. “Effects of Active Music Therapy on The Normal Brain: Fmri Based Evidence.” *Brain Imaging and Behavior*, Springer US, 7 Apr. 2015, https://link.springer.com/article/10.1007/s11682-015-9380-x.
25. Rai, Ankit. “Python: CV2 Canny() Method.” *Java2Blog*, 23 Dec. 2020, https://java2blog.com/cv2-canny-python/.
26. “Ratios and Musical Intervals - Math.wustl.edu.” *Ratios and Musical Intervals*, Washington University in St. Louis, https://www.math.wustl.edu/~wright/Math109/M&MCh04-07.pdf.
27. Rosebrock, Adrian. “OpenCV Edge Detection (cv2.Canny).” *PyImageSearch*, 9 May 2021, https://www.pyimagesearch.com/2021/05/12/opencv-edge-detection-cv2-canny/.
28. Sam, N. “What Is Medial Prefrontal Cortex? Definition of Medial Prefrontal Cortex (Psychology Dictionary).” *Psychology Dictionary*, 7 Apr. 2013, https://psychologydictionary.org/medial-prefrontal-cortex/.
29. Sanderson, Grant, director. *Music and Measure Theory*, 3blue1brown, 4 Oct. 2015, https://www.youtube.com/watch?v=cyW5z-M2yzw. Accessed 29 Nov. 2021.
30. Sharma, Samata R, and David Silbersweig. “Setting the Stage: Neurobiological Effects of Music on the ...” *Neurobiological Effects of Music on the Brain*, Berklee Music and Health Institute, 1 June 2018, https://remix.berklee.edu/cgi/viewcontent.cgi?article=1005&context=mh-exchange-music-medicine.
31. Sihvonen AJ;Särkämö T;Leo V;Tervaniemi M;Altenmüller E;Soinila S; “Music-Based Interventions in Neurological Rehabilitation.” *The Lancet. Neurology*, U.S. National Library of Medicine, https://pubmed.ncbi.nlm.nih.gov/28663005/.
32. Skouras, Stavros, et al. “Superficial Amygdala and Hippocampal Activity during Affective Music Listening Observed at 3 T but Not 1.5 T Fmri.” *NeuroImage*, Academic Press, 12 July 2014, https://www.sciencedirect.com/science/article/abs/pii/S1053811914005795?via%3Dihub.
33. Trimble, Michael, and Dale Hesdorffer. “Music and the Brain: The Neuroscience of Music and Musical Appreciation.” *BJPsych International*, The Royal College of Psychiatrists, 1 May 2017, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5618809/.
34. Tsujii T;Sakatani K;Masuda S;Akiyama T;Watanabe S; “Evaluating the Roles of the Inferior Frontal Gyrus and Superior Parietal Lobule in Deductive Reasoning: An RTMS Study.” *NeuroImage*, U.S. National Library of Medicine, https://pubmed.ncbi.nlm.nih.gov/21749923/.
35. User, 6613600. “What Are the Correct Usage/Parameter Values for Houghcircles in Opencv for Iris Detection?” *Stack Overflow*, 1 July 1960, https://stackoverflow.com/questions/10716464/what-are-the-correct-usage-parameter-values-for-houghcircles-in-opencv-for-iris.
36. Wikipedia, Organization. “Inferior Frontal Gyrus.” *Wikipedia*, Wikimedia Foundation, 16 May 2021, https://en.wikipedia.org/wiki/Inferior_frontal_gyrus.

Acknowledgement

Sincere thanks to the faculty member Radhakrishna Madas from On My Own Technology Private Limited’ who guided and mentored me in the project.

A special mention to Ms. Reetu Jain Chief -Mentor On My Own Technology who gave special insights for collecting data and conducting the detailed analysis.

Authors

First Author – Siddh Merchant, Podar International School, siddh.merch@gmail.com

Second Author – Reetu Jain, Chief Mentor on My Own Technology Pvt Ltd. reetu.jain@onmyowntechnology.com