Xylobot: How a closed-loop system plays an instrument

Maastricht University, Department of Data Science and Knowledge Engineering

Ewan Demeur (i6155238), Rik Dijkstra (i6135069), Rhys Evans (i6150369), Kailhan Hokstam (i6154002), Valentin Maican (i6150862), Benjamin Rodrigues de Miranda (i6152983), Cas Teeuwen (i6169583)

***Abstract*—In this paper we discuss and explore the creation of a xylophone player – a robotic arm with a mallet that can be programmed to play a xylophone, named and referred to as "Xylobot". Sound processing is used to listen to a song played by a human operator. The robotic arm can then replay the song, and even improvise a further sequence of notes based on a statistical model of the song.**

Keywords: ***xylophone, music robot, music AI***

I. Introduction

Playing musical instruments is a talent of the human species. Like most other skills, there is an interest in automating it. In this case, there is a one-armed servo-controlled robot playing the xylophone, using sound detection and computer vision. The robot makes use of an AI for playing notes, based on what a human played before it. The aim of this paper is to discover the optimum configuration for the closed-loop system. Optimum, in this case, is defined by the configuration that plays the xylophone in the most enjoyable manner. In order to reach an answer to our problem statement, a few research questions were constructed. In what way can the robot be controlled, to yield the most accurate results at any given tempo? It is believed that the triangular hit method is most accurate, because it minimizes shaking. A slower pace and a bigger delay will yield more accurate results. What window size and hit detection thresholds gain the best performance in the Short-Term Fast Fourier Transform? It is known that higher thresholds will most likely result into more false negatives and less true positives, but where the ideal threshold lies remains to be explored. What effect does increasing the probability of rare note combinations occurring in an improvised melody have on the similarity between the improvised piece and the original? In principal, increasing the chance of note couples which are not seen in the original piece appearing in the improvised piece should cause the the improvised melody to be less similar to the original melody because new, unseen note combinations will come up more frequency.

At this point, many musical robots have been developed. One can see that the goal and design of each xylophone-playing robot is very different. Narrowing down the research to robots that operate with multiple (connected) rotating joints, and a maximum of two mallets, we are left with only a few pieces of research. In the first of these, a pre-manufactured 'NAO' robot is used, which plays xylophone with the goal of increasing an autistic childs' performance at the instrument [1]. It uses a Microsoft Kinect for the capturing of the child's hits. It also makes use of a Fast-Fourier Transform for that same purpose. A group of students from Stanford have used an industry-grade robot for playing the xylophone. It functions by first detecting what the user played, and then replicating that sequence [2].

II. Interface and Artificial Intelligence

A. Graphical User Interface (GUI)

The robot is synchronized with a GUI panel which displays graphics that display various aspects of the processes taking place. Two of the five sections are taken up by a simulation showing both a birds eye view and a side view of the robot's actions. The third panel shows real time signal processing through graphical means, and the fourth shows the current image captured by the computer vision. Finally there is a large fifth section which allows the manual playing of the xylophone without the actual robot.

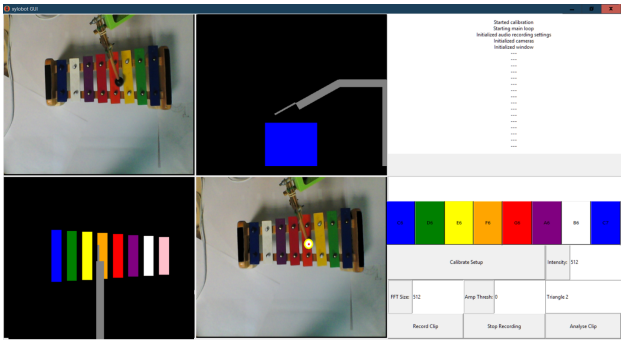

Figure 1: GUI Example

B. Simulation

Python library TKinter was used to simulate the robot arm and the xylophone [3]. All robot arm segments and keys are initially drawn onto two canvases, one canvas has a birds eye view and the other has a side view. The canvases are updated simultaneously when a new goal position or new joint angles are provided. Forward kinematics is achieved by summing the x and y coordinates of the robot arm endpoints to calculate the next components endpoint coordinates. All joint angles are moved one degree each closer to their goal angle each

iteration of a loop, after which the thread sleeps for a time determined by how long the entire movement should take. Backwards kinematics is achieved by the way of the FABRIK method [4]. Additionally, the simulation represents how the xylophone is placed. It is placed in the right position with the right rotation after the robot is calibrated using the computer vision.

C. Artificial Intelligence

One of the signs of a competent musician is the ability to improvise on a theme. It is possible to mimic this with the use of Markov chains. This is a probability based approach which learns how to play based on a sequence of notes performed by the user.

1) Improvisation

Each note in the improvised section depends only on the note before it. The decision of which note should follow another is made with the aid of a transition matrix, which is a structure showing the probabilities of going to each possible note from a given base note. The matrix is constructed in such a way as to encourage the improvised section to link notes which appear adjacently in the music provided by the user [5]. Frequencies of note couples are counted in the provided note sequence and then the actual matrix entries are generated by calculating Laplace estimates with these frequencies. Using raw probabilities would leave certain transitions with a pro ability of zero, which would limit the freedom of the robot to create unique music. Laplace estimates assign a small non-zero probability to each transition which would otherwise not be possible [6]. For the sake of scalability, note couples are handled by a python dictionary which is computationally faster than a tediously long conditional statement. While this may not make a significant difference in the current conditions, when longer sequences of notes, or when more notes are available, the use of a dictionary will be very beneficial.

Although the pitch of the notes played make up a large part of the music, timing is also a major factor. Instead of keeping notes evenly spaced throughout the piece, the timing is varied. A probability based approach yields an improvised section which fits the provided piece. This is because the probabilities of the note timings are based on the original music. The probabilities here represent the likelihood of a time delay occurring between two notes being played. Before generating these probabilities, the original music upon which the timings are based must be quantized so that their timings are all multiples of a constant interval as described below. This makes it possible to count how many notes are played with each possible time delay from a previous note, something which would not be possible if the notes were still based on a continuous timeline. Using these frequencies from the original piece, the probability of each note timing occurring in the new section can be calculated. The first note of the improvised section is based on the last note of the original section, and all of the following notes are based on the previous improvised note.

Adjusting the notes to fit a rhythm is done by creating a quantized timeline upon which notes are fitted [7]. The interval between is determined by the smallest time separation between any two recorded notes in the provided music. Then the rest of the notes have their timings adjusted to fit the nearest quantized point on the timeline.

The advantage of using Markov chains is that the improvised piece is created based solely on the piece played by the user, ensuring that it fits well in the overall music [5]. Additionally, the use of Laplace estimates allows the algorithm some creative freedom to play notes which are unexpected, but with a low enough probability that they do not occur frequently enough to change the underlying theme of the music. By quantizing the music the notes are fitted to more stable rhythm, but due to human error can not be timed perfectly. Furthermore, the quantization of the notes allows time delays between pairs of notes to be counted so that a consistent rhythm can be used in the improvisation.

A shortcoming of using Markov chains is that the piece of music provided must be at least twenty notes (as a guideline but it depends on the piece of music) before the Laplace estimates for note transitions which did not occur in the original piece become small enough (less than $\frac{1}{8}$) to be considered unlikely. This is based on the following equation for Laplace estimates shown in Eq: 1:

Laplace Estimates

$$\frac{(f_{ij}+1)}{(\sum_i (f_{ij})+n)} \qquad \text{(Eq: 1)}$$

where f_{ij} is the frequency of the transfer between note i and note j, and n is the number of notes possible.

This equation is based on the general formula for Laplace estimates[6]. The reason that long sequences reduce the probability of unseen transitions occurring is because with more notes, transitions from one note to another will occur more often if they work well together but odd transitions will not take place so their frequency will remain low. The summed term in the equation represents the total sum of the transition frequencies from one note to all other notes. With more notes this summed term increases while the frequency of unlikely transitions does not increase, thus increasing the denominator while the numerator stays the same. A disadvantage of the quantization algorithm is that it assumes the user has a basic understanding of rhythm, which may not be the case when young children use the xylophone. This presents an issue because the shortest interval will not be representative of the timing of the rest of the piece, that is, the other intervals will not be multiples of this smallest interval, potentially causing the timing scheme to be interpreted by the computer in a way which was not intended by the user. This could also happen when the shortest time interval is too long, as this means the rest of the notes will shift by a larger amount because the points on the timeline are further apart.

III. Robot Control

The robot to be used for the duration of the project has been 3D printed in the laboratory, is using Arduino for control and communication and consists of 3 Servomotors with a total of

3 degrees of freedom. Taking the human arm as an analogy for our robot arm, for an easier visualization of it, the following names will be used for the robot parts containing each of the Servomotors:

- Shoulder - the base motor (m_0), with a length of 18.8 cm (our origin height for any further calculations will start from this point), has 1 degree of freedom, allowing for a yaw rotation (around z - the height axis), is in charge of rotating the robot, so that it moves the mallet between keys
- Elbow - the middle motor (m_1), with a length of 10.55 cm, has 1 degree of freedom, allowing it to rotate a maximum of 180 degrees around the x-axis (parallel to a xylophone placed in the default position)
- Wrist - the top motor (m_2), with a length of 5 cm, has a mallet of 13.7 cm attached to it, has 1 degree of freedom so it can rotate 180 degrees around the x-axis, and, together with the elbow, is responsible of reaching and hitting the xylophone key

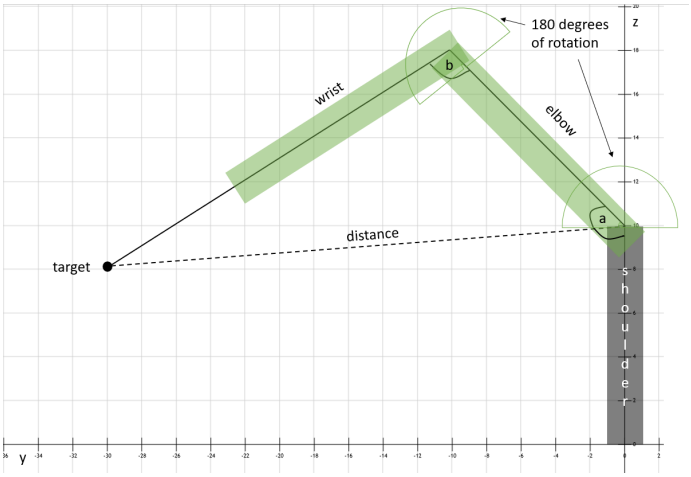

Figure 2: Robot Side-view

The communication with the robot is done through Arduino, the data passed to it being a tuple with 3 angle values, one for each motor. From within the program, Python's 'Serial' library was used in order to send those values to the Arduino software. It is important to ensure that the angles sent can not potentially damage the robot, for example by forcing it to rotate beyond the physical boundaries of 180 degrees. In order to do that, a safety layer has been implemented, which would check the values and make sure they're within the allowed rotational ranges before sending them. The safety layer allows for experimentation and creation of kinematics methods without fear of damaging the robotic setup.

A. Forward Kinematics

The first movement trials were done using Forward Kinematics, meaning the calculations required to reach a given target were done directly using motor angles. For that, a hard-coded list of motor angles (one for each note of the xylophone in its default position), was used. Given a note value, the program would look up the values and then calculate the path (consisting of angle tuples) required to get there. The calculations were done by projecting the motors on a 2D space and gradually translating them till the target angle tuple was reached. While effective, forward kinematics requires hard coded angles that cannot account for movement of the xylophone.

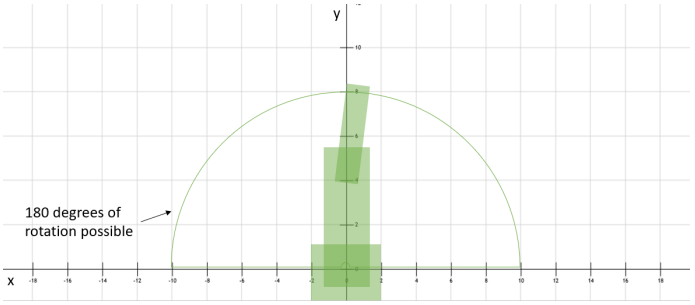

Figure 3: Robot Top-view

B. Inverse Kinematics

With the help of inverse kinematics, it is possible to get real life coordinates (calculated in cm) and transform them into motor angles, making it easier to perform all the required calculations. The first step that needs to be done is calculating the distance from the determined origin (the base of the robot), to the target as shown in Eq: 2 and Figure 2:

Distance from the base of the robot to the target

$$d = \sqrt{(y_{target} - y_{origin})^2 + (z_{target} - z_{origin})^2} \quad \text{(Eq: 2)}$$

where d is the distance from our origin to the target, y_{target} and y_{origin} are the y-coordinates of the target and origin, z_{target} and z_{origin} are the z-coordinates of the target and origin.

Once we know the distancefrom the origin to the target, we can calculate the angles between all the servo motors. The angle between the wrist and the elbow can be calculated using the Law of Cosines in Eq: 3. Once we know the cosine of the angle, we can calculate the angle itself by taking the inverse cosine of it, and for our further calculations we'll transform it into degrees as also shown in Eq: 3.

Law of Cosines & Wrist-Elbow Angle Calculation

$$cos(b) = \frac{{l_{elbow}}^2 + {l_{wrist}}^2 - {d_{target}}^2}{2 \cdot l_{wrist} \cdot l_{elbow}}$$

$$b = acos(cos(b)) \cdot \frac{180}{\pi} \quad \text{(Eq: 3)}$$

where b is the angle between wrist and elbow, l_{elbow} and l_{wrist} are the lengths of the elbow and the wrist, and d_{target} is the distance to the target.

Similarly, we apply these methods to calculate the angle between the elbow and the shoulder. The only difference being that we need to perform two calculations and then sum up the angles: Eq: 4

Elbow-Shoulder Angle Calculation

$$cos(a_1) = \frac{{l_{elbow}}^2 + {d_{target}}^2 - {l_{wrist}}^2}{2 \cdot d_{target} \cdot l_{elbow}}$$

$$a_1 = acos(cos(a_1)) \cdot \frac{180}{\pi}$$

$$a_2 = asin(\frac{y_{target}}{d_{target}}) \cdot \frac{180}{\pi}$$

$$a = a_1 + a_2$$

(Eq: 4)

where a is the final value of the angle between the elbow and shoulder, a_1 and a_2 are the two angles to be calculated and summed in order to get the final value, l_{elbow} and l_{wrist} are the lengths of the elbow and wrist, d_{target} is the distance to the target, and y_{target} is the y coordinate of the target.

In order to find the angle of rotation (done at the shoulder), the inverse tangent of the target coordinates is calculated, as shown in Eq: 5:

Base angle calculation

$$c = atan(\frac{x_{target}}{y_{target}}) \cdot \frac{180}{\pi}$$

(Eq: 5)

where c is the angle of rotation at the base (shoulder), x_{target} and y_{target} are the x and y-coordinates of the target.

Now able to get the right motor angles for any point on our xyz space, corresponding to real centimeters, it is possible to perform the movement from any position to our target by calculating a path of points between them. Different ways of calculating that path described as hit methods.

C. Hit Methods

Different hit methods have been implemented, one reason for that being the possibility to do various experiments and see which hit method is optimal, as well as which of them works best in specific situations. These methods generate a path of points between the current and target positions, the scarcity of the points being dependant on the speed of the hit and the distance travelled being related to the power required. There's a separate way of hitting the same key over and over, so a specific method had to be used. In order to do this, we keep the x and y coordinates of the robot constant, and, depending on the speed and power given, raise the robot hand up to the z_{target} height and then perform the hit. For the cases where the bot is travelling from note to note (which is most of the time), four ways of generating the path points necessary to reach the target, have been implemented:

- *Quadratic* Given two points on a 2D space representing the x, y_{origin} and x, y_{target} coordinates, and a midpoint $z_{midpoint}$ representing the height at the middle of the motion, it is possible to fit a quadratic equation between these three points, and then split it based on the given speed, generating the required path. First, an offset function based on the given points is visualized, starting at 0. Then the $ax^2 + bx + c$ equation is used - knowing the inputs and outputs (x and y of the points given to us), we can replace the x in the quadratic equation with the input, and find out the a, b and c coefficients with the help of the output.

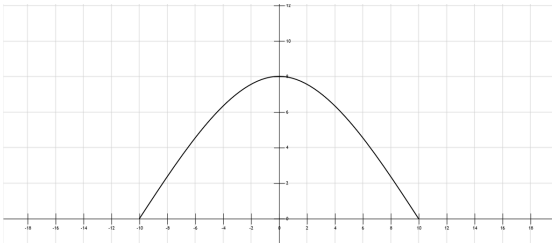

Figure 4: Quadratic Hit

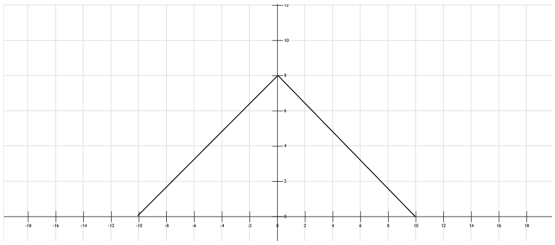

Figure 5: Triangular 1 Hit

- *Triangular 1* Given two points on the plane, and a calculated midpoint between at the height required to hit, raise the mallet on the slope between the midpoint and the origin. Once at the top, perform the hit. All the information required in order to calculate the slope and get the linear equation is available. Once the equation is calculated, move on the x-axis (width) and record the values of the z-axis (height). The hit in this case comes at an angle.

- *Triangular 2* similarly to the first triangular method, we determine the linear equation $y = mx + b$ by finding out the slope and the intercept given the origin and target points, the only difference being that no midpoint is required, the mallet travelling all the way until it reaches the height of the target point, and performs a downwards hit.
- *Uniform* To perform a uniform hit, the mallet goes in an upwards motion from the starting point until it reaches the height necessary for the hit, then travels in a straight line till it reaches the x and y coordinates of the target,

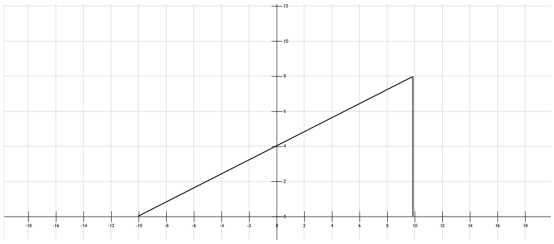

Figure 6: Triangular 2 Hit

and hits it in a downwards motion. The calculations are straightforward, as it is a constant function.

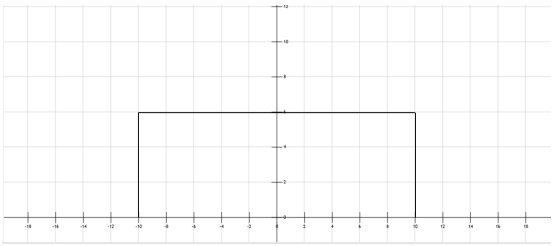

Figure 7: Uniform hit

In practice, the quadratic hit wasn't used as much due to it being less effective than the triangular one (higher amount of calculations, harder to implement a clean hit), so the experiments and general musical practice have been done using the triangular (1, 2) and uniform hits.
With a produced list of points that would perform a hit given an origin and target, it's possible to change the speed and power of the hits to affect the sound created by a hit.

D. Velocity & Power Control

When deciding how to perform a hit, there are certain aspects that have to be considered. The tempo of the song (i.e. how quickly notes have to be hit sequentially), the distance between two given notes and the velocity of a hit (i.e. note dynamics e.g. 'p' - piano, 'f' - forte). Obviously, pace and power play a major role in how the movement is to be done, thus some codependency rules had to be thought of:

1) Speed will be considered in amount of movements between the origin and target, that amount being dependant on the tempo of the song, and one movement being roughly equal to the distance between the xylophone keys (approx. 3 cm).
2) The maximum speed would be reaching the target in one move, with a 2nd move producing the hit.
3) The bigger the tempo value, the more subdivisions are to be used when calculating the path, producing more path points and decreasing the speed.
4) Power will be a coefficient added to the standard height of a xylophone key, resulting in a value telling the robot at what altitude it should stop before hitting. The higher the resulting height, the more powerful the hit.
5) Power is affected by a bounce coefficient, which is altering the default height of the key, in order to get a smoother tone when hitting the note by diminishing the bouncing effect of the mallet.
6) Power is in relationship with the distance to the target, being necessary to decrease or increase it at some points in order to perform the hit in time.

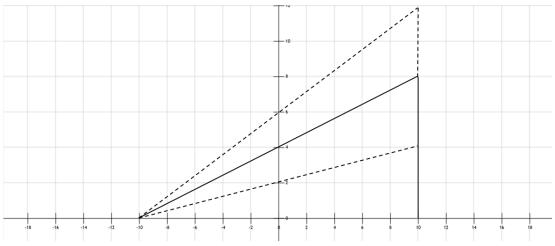

Figure 8: Power difference

7) The used power values are:
   - 'pp' - 'pianissimo' - very quiet = 1
   - 'p' - 'piano' - quiet = 2
   - 'mp' - 'mezzo piano' - moderately quiet = 3
   - 'mf' - 'mezzo forte' - moderately loud = 4
   - 'f' - 'forte' - loud = 5
   - 'ff' - 'fortissimo' - very loud = 6

Considering all of the above, in order to get the speed, the distance (with regard of the power), is divided by the sum of the distance between the keys and the tempo.

E. Class design

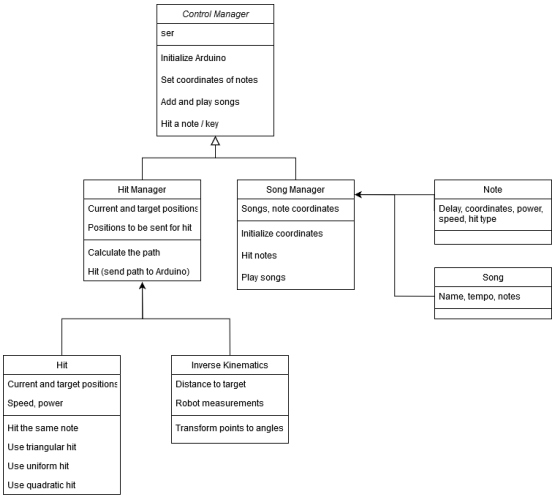

Figure 9: Control Class UML

Combining all of the control components is necessary to easily integrate other parts in the program. The control code also needs to be able to be tweaked without having major consequences for the rest of the code. To do this, a design pattern needs to be applied. Facade was the design pattern chosen in this case, as it limits the communication of the rest of program's components, with the control components, to a class named 'ControlManager', which will be referred to as (CM) for short. (CM) communicates with the 'SongManager' (SM) and the 'HitManager' (HM). (CM) provides the other managers with the information required to perform the control

tasks, while (HM) does the actual movement calculations and sends them to the robot, and (SM) is responsible of storing and playing songs added by the user.
The Hit Manager delegates the tasks to the 'Hit' and 'Inverse Kinematics' classes. 'Hit' contains the algorithms that return the path of points in order to move from the origin to the target, based on the hit method chosen. 'Inverse Kinematics' is responsible for analytically transforming the real life coordinates to motor angles, as well as check that the resulting angles are not going to damage the robot.
Using the above pattern assures low coupling and high cohesion for the inter-class communication, allowing for increased readability, change flexibility and possibly performance.

IV. Calibration through Computer Vision

A. Computer Vision Key Identification

For the robotic arm to be able to hit the keys and produce clear notes, the arm must know where each key is. Identifying the xylophone in the real world was achieved through the use of computer vision. The openCV package was used to complete the task [8].

1) Detecting Keys Through Colour Detection

Detecting the the keys is attempted using colour detection. After trying colour detection in the RGB and HSV colour spaces, it seems that the HSV colour space is more effective at identifying colours when lighting conditions changed, thus the HSV colour space is used in the method. Individual keys are identified using a colour range that matches the colour of the key. Not every key can be identified due to reflections and incorrect identification. One major shortcoming with the colour detection is that any reflection on the keys makes it impossible for the program to identify the keys as the reflection is identified as white by the colour detection program. Thus the challenge is to find the positions of the keys that cannot be found using the colour detection. This problem is solved by identifying the keys that can be found and then calculating the positions of the remaining five keys using the positions and sizes of the keys already identified.

Figure 10: Colour Detection Results

In Figure 10 this method successfully calculates the midpoints of all the keys to an acceptable degree of accuracy, however when the setup moves to a different location with a different lighting, the keys that are detected correctly in Figure 10 will not be detected well in the new lighting due to reflections or changes to the colour profile. Because of this, the results become much less accurate, as can be seen in Figure 11. Colour detection to identify keys gives positive results in the right conditions, however its inability to adapt to different lighting conditions affects its accuracy.

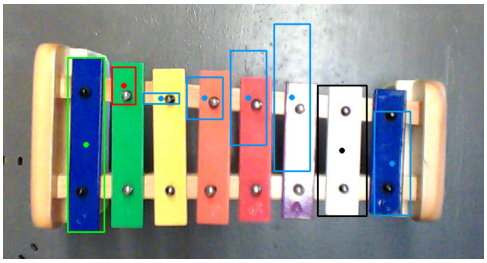

Figure 11: Colour Detection Failure

2) Template Matching with Edge Detection

The use of edge detection to match a template of the xylophone to the live image is also explored. The canny algorithm is used to detect the edges in the image frame, which is then used to identify the xylophone and thus the individual keys [9]. To do this the canny detected edges are matched to a premade template using template matching. Multiple templates have been tested including a screenshot of an optimal canny mask. This does not get results that accurately matched to the correct part of the image. Also the matching is hindered further by the limitations of the edge detection. Just like the color detection, the edge detection output changes with different lighting, thus making it difficult to find an ideal template to match to the live image.

3) Identifying the Sides of the Xylophone

To combat reflections on the xylophone from the light source, black tape was taped to the sides of the xylophone and was painted with matte black paint. The matte paint reduces reflections, and in testing, it is seen that black is the color that is detected more effectively, as it absorbs the most light and thus does not change heavily in the HSV colour space when the lighting changes. The two sides are then identified through colour detection, and ellipses are fitted to the area of tape [10]. The mid points of the two ellipses are used to draw a line across the xylophone. The midpoint of each key is mapped on the line. Because the mallet only needs to hit the middle of each key, the midpoints are all that is needed. To ensure that a false line is not created, the ellipses identified are bound to a certain size. Ellipses are used as these shapes can be fit to a mask at an angle thus allowing for the xylophone to be detected at any orientation. The result of this method can be seen in Figure 12. This method is able to accurately identify the midpoint of each key, and is less susceptible to reflection and changed lighting conditions.

B. Computer Vision Mallet Tracking

To calibrate the setup, the mallet is tracked as it moves over the different keys. Color detection in the HSV color space with size restrictions is used to identify the black head of the mallet during this process. Size restrictions are added to the area and location of the mallet head to make sure it is not falsely identified in an incorrect location. HSV was chosen for the color space because during our preliminary testing, we

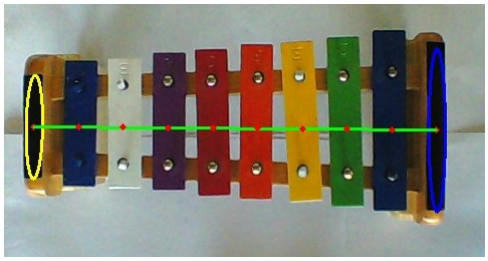

Figure 12: Ellipse Fitted

found that is was easier to identify between the blue keys and the black mallet in HSV space. In RGB space these two areas were often identified as one when shadows made the blue keys appear darker. However, this was mitigated in HSV.

To calibrate the setup and find the centerpoints of each key, the pixel coordinates of the mallet head are compared to the pixel coordinates of the key's midpoints that were found in the key identification step. The robot moves the mallet in small increments to match these pixel coordinates and then saves the mallet coordinate in the robot space for every key. These coordinates are used by the robot to return to each key while playing the xylophone. Extra measures were implemented to increase calibration success rate such as decreasing the step size every step a key is not found and also moving the mallet to the previous key when the mallet is not found. An example of the mallet tracking is shown in Figure 13.

Figure 13: Mallet Tracking Example

C. Adding CV to the Closed Loop

Having integrated the computer vision, it is now possible to consider it in the closed loop of our program, its function being of constantly monitoring the position of the xylophone, help calibrate the robot thus producing a new set of note coordinates, which are to be used for performing the hits. In the case the xylophone is moved from its default position, computer vision will take note of it and go through the calibration process again, so that the robot can hit based on the new position. The general closed loop can be seen in Figure 14.

V. Sound Processing

To enable the Xylobot to replay notes played by a human on its xylophone, sound processing is used to identify which, when and with what intensity keys are hit.

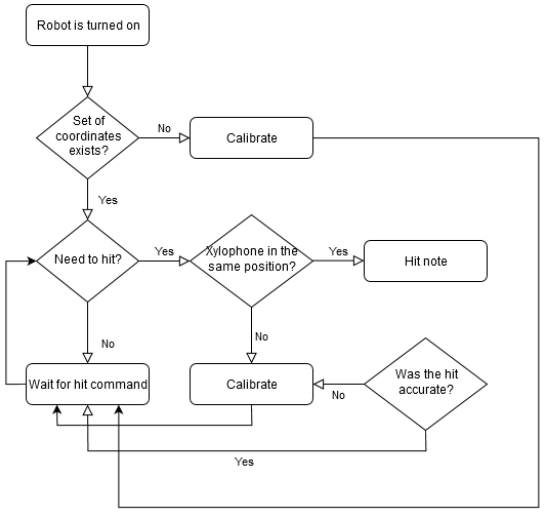

Figure 14: Closed loop flow chart

To detect which key is hit, the concept of a short-time Fourier transform (STFT) [11] is used to utilize the Cooley-Tukey, vectorized, non-recursive fast Fourier transform (FFT) [12] to calculate the discrete Fourier transform (DFT) [13] over time of a digital signal recorded by a portable USB condenser microphone placed under the xylophone. This makes it possible to examine the power of sets of frequencies for moments in time also referred to as time frames. From testing using the Android applications gStrings Tuner[14] and Pitched Tuner[15] it is known that the keys of the xylophone range from C_6 to C_7 almost following standard tuning frequency as set by ISO 16:1975, the specific frequency, associated with each key of the specific xylophone used can be found in appendix A, the mean of the applications readings are taken taken as baseline frequencies. These frequencies can then be associated with one of the eight keys on the xylophone, based on the assumption that the bin with the most power contains the fundamental frequency (Gerhard, 2003)[16].

For detecting when a note is hit all local maximal peaks of the mean of the magnitudes of the frequency bins gotten from the DFT of the recorded signal are found by simple comparison of neighbouring values using SciPy[17] and based on if these local peaks have a topographic prominence, which, as explained by Griffié et al (2015)[18] is a peak its relative height above the lowest contour that surrounds it without encircling any higher peak, above a certain threshold they are identified as moments in time at which a hit took place. This prominence threshold is based on the maximum magnitude of the recorded signal divided by a sensitivity factor. Because of the fact that a xylophone is a (pitched) percussive instrument with thus a near immediate maximum amplitude after triggering a sound (Blades, 1992)[19] these peaks give a reasonable approximation of the moment a key is struck by the mallet.

The intensity of a hit is found by taking the mean of the magnitudes of frequency bins gotten by the DFT of the recorded signal at the moment in time at which the corresponding peak is detected.

Expanding on the workings of the implementation of key

detection using sound, the STFT using FFT for DFT of sound is accomplished using several Python packages. PyAudio[20] is used to record and save WAV files and SciPy[17] to load WAV files, an audio file format standard (Luo, 2012)[21]. The STFT can be thought of as performing a FFT on a part of the signal contained in a window then sliding this window across the signal. Then to perform STFT certain parameters need to be set most importantly, the window length. As explained by Czerwinski & Jones (1997)[22] a shorter window length allows for greater time resolution, while a longer window length allows for greater frequency resolution. Because the STFT needs not more frequency resolution then to distinguish keys, a relatively shorter window length is sufficient, which would allow to more accurately determine when the key is hit than when using a larger window length. The window type chosen is a Blackman window, which according to Podder (2014)[23] is superior to other popular windows, specifically the Hamming and Hanning windows in various use cases, among which their responses to multiple filters with different parameters in MATLAB simulations. The Kaiser window can perform better than a Blackman window (Muralidhar, 2010)[24], but it requires an extra parameter to tweak and is therefore not explored due to time constraints. The overlap length is set as half the window length leading to a hop length equal to the overlap length. This ensures that the sum of the area of the windows taken is 1, leading to the amount of energy represented by the amplitude of the signal staying the same when comparing the original and transformed signal.

In exploratory testing it is found that allowing Xylobot to find the pitch of hit in a time period starting from a time frame it detects a hit increases performance. The exact reason for this is not known but it is hypothesized that at a frame seen as a peak there are objects that are not the key creating sound, heightening the amount of energy and thus peak in signal and particularly creating unwanted noise. Xylobot then analyzes the frequency spectrum at the next time frame to see if it recognizes a frequency associated with a key until a key is found or a certain time threshold is met, set at 20 milliseconds. This period has not further been tested due to time constraints, however a closed hi-hat off by 20.8 milliseconds in a sequence of 16 beats seems not noticeable in uncontrolled experiment with 180 beats per minute, while an offset of 41.6 milliseconds is, potentially indicating that while this time period might contain another hit and detect the pitch of this hit instead of the original hit, this subsequent hit is not likely to be registered as a separate hit in the first place by the human ear, helped by the longer decay of a xylophone key in comparison to a closed hi-hat.

VI. Experiments

A. Control

With multiple options for controlling the setup, experiments would show which method is most accurate at all speeds. For this most accurate method experiments would show at what speeds it is most proficient, meaning making less than one mistake per sequence.

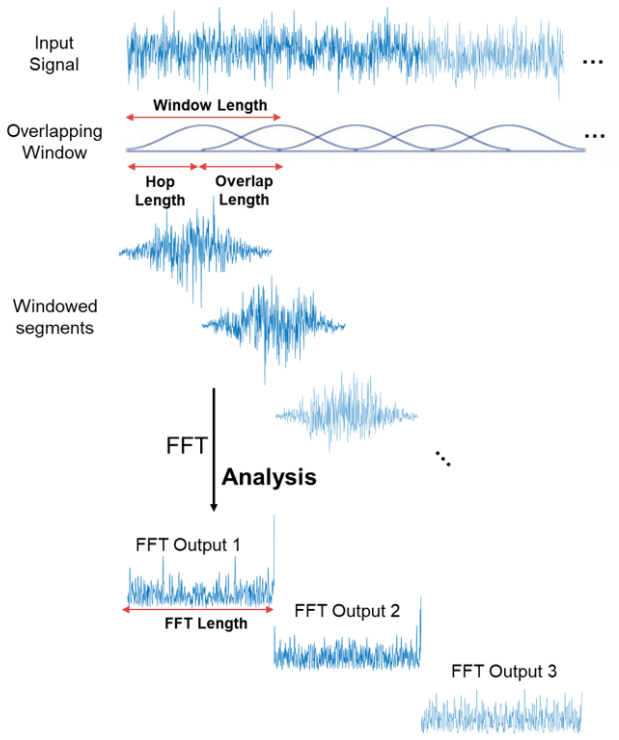

Figure 15: STFT Window illustration from MATLAB [25]

1) Triangle 1 and Triangle 2

From òbservations we concluded that the quadratic hit method did not perform better than the triangular hit methods. The uniform hit seemed to perform worse and at the least move unnaturally. Since we had to prioritize what to test we decided to test both triangular methods. In this experiment, we run the same test sequence of 20 notes 20 times for each tempo and method and counted the imperfect hits. Imperfect hits can be anything from a double hit to hitting two keys or not hitting a key at all. The tempo stands for the amount of intermediary locations fed to the setup. This controls the speed. Tempo zero sets only three points and tempo one sets four, etcetera.

2) Tempo and accuracy

Continuing with the method found to be the most accurate, we test at what speeds the setup will be enjoyable to listen to. We define this to be at less than one mistake in the test sequence of 20 notes. This sequence is meant to be difficult for the robot and it is therefore to be expected that it will not always get each note.

B. Similarity of Improvised Melody to Original

The aim of this experiment is to mimic the creative freedom given to an artist when improvising by allowing the robot to play notes which deviate from the original theme, and measure the effect of this on how dissimilar the improvised piece becomes compared to the original. This is achieved by increasing the probability of note transitions which were not

seen in the original piece. Thus the following question is of interest her: how does increasing the probability of rare note combinations occurring in an improvised melody affect the similarity between the improvised piece and the original should be investigate?

”It is not immediately clear how to obtain a reference ground truth for music similarity, since it is a naturally subjective phenomenon”, but the approach created for this experiment is one which focuses on a combination between distance between notes and chord structure. [26] Distance is measured between the improvised piece as the sum of the distances between the two notes (one from each sequence) in each position. The distance between a single pair of notes is simply the number of half steps between them [27]. A score is assigned to a pair of sequences which is initially zero, but increases with each note distance. Thus a higher score indicates more dissimilarity between two sequences. The score is reduced each time a note in the improvised sequence is one of the notes of the notes in the base triad of the key upon which the music is based. This is because major triads contain only notes which will always sound good when played as part of the melody, so when a note in the improvised piece falls within this category it will always tend to fit with the original piece as the original is in the same key. [28] Each melody used in this experiment is based in C major, so the notes counted in the triad are c6, e6, and g6. Finally, the score is summed for each possible alignment of the two sequences. This is because the distances between notes are being considered, not the order of the sequence, so each note of the first sequence must be measured against each note of the second sequence.

Increasing the probability of unseen transitions can be taken to mean increasing the change of a note couple being played which is not seen in the original piece. The probability is increased from 0.00 to 0.04 in 5 steps. For each note, if no transition between it and another note occurs in the original piece, then the increase is applied on top of its Laplace transform to increase its probability of occurring in the improvised sequence. To ensure that the probabilities of a notes transitions still sum to 1, the overshoot caused by increasing the unseen transition probabilities is compensated for by reducing the probabilities of the other transitions occurring. Similarity scores are then calculated for each increase increment 5 times and averaged out. This is done with three different melodies of increasing lengths.

In principal, increasing the chance of note couples which are not seen in the original piece appearing in the improvised piece should cause the the improvised melody to be less similar to the original melody because new note combinations will come up more frequency.

C. Optimizing Sound Processing

The sound processing process involves multiple parameters that can be tweaked. To minimize the amount of misidentified keys an experiment has been set up to identify optimized parameters. For this experiment a human hits the 8 keys of the xylophone in a slow and a fast manner. The guideline for playing in a slow manner is taken as waiting until the previous sound has completely decayed. With the fast manner the human tries to hit the keys as soon as possible after each other, while trying to hit cleanly, emulating someone playing the xylophone in a real life scenario. The keys are hit 5 times from the lowest key C_6 to the highest key C_7, and 5 times from the highest to the lowest key for the slow and fast manner. This then, in turn, leads to 10 recordings in a slow manner and 10 in a fast manner and 20 total recordings.

The actual parameters changed are the FFT window size, for which the values 128, 256, 512, 1024, 2048 have been used and the hit detection threshold values are 0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. All combinations of these values have been used and these led to our results, which can be seen as quite bad, because non-optimal combinations are included in calculations, noticed in the averages for example. Also due to the fact that a human played the recorded samples the exact timings of supposed hits are not known. To ensure that no false positives are encountered with regards to identified keys, the supposed sequence of keys and the sequence resulting from the sound processing are strictly compared. They need to be in the exact same order and as a result as soon as a note is inserted or deleted by the sound processing that is not supposed to be there, this has a large impact on the subsequent part of the sequence, which is then likely to not match.

VII. Results

A. Triangle 1 and Triangle 2

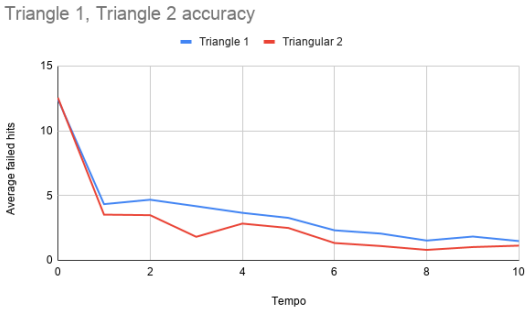

Figure 16: Experiment 1

B. Tempo and accuracy

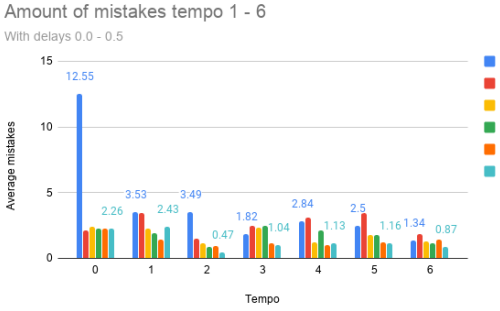

Figure 17: Experiment 2

In Figure 17 the average amount of mistakes are shown for each tempo and delay. Delay is the amount of seconds the setup waits in between playing notes. It becomes apparent that, as expected, the setup works better at lower speeds and with more intervals.

C.

	Slow (Mean)	Slow (SD)	Fast (Mean)	Fast (SD)	Two-sample t-test
Misidentified keys	3.958	3.284	5.855	2.6681	1
Flatness	0.01842	0.006820	0.002215	0.001006	1
Effective Duration	15.5005	6.4866	1.4234	1.0017	1

Table I: Slow (Mean), Slow (SD), Fast (Mean), Fast (SD), and Two-sample t-test Results

	Slow (Mean)	Slow (SD)	Fast (Mean)	Fast (SD)	Two-sample t-test
Misidentified keys	0.2	0.6325	3	3.1269	1

Table II: Misidentified keys with optimal parameters, FFT window size = 2048, Hit detection threshold = 0.34

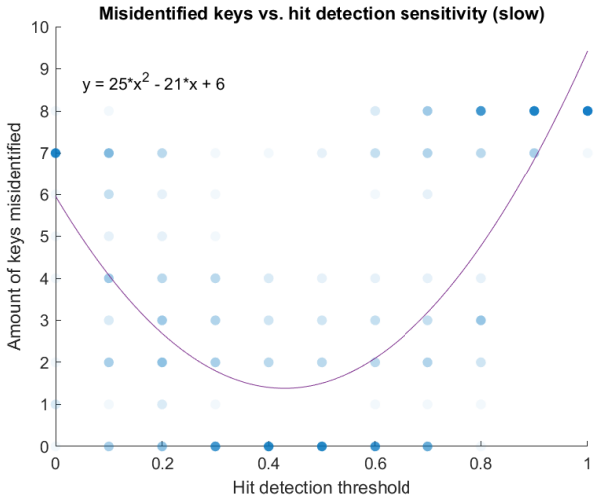

Figure 18: Scatter plot with quadratic trend line showing relation between misidentified keys and hit detection sensitivity, minimum of trend line is at x = 21/50, playing in slow manner

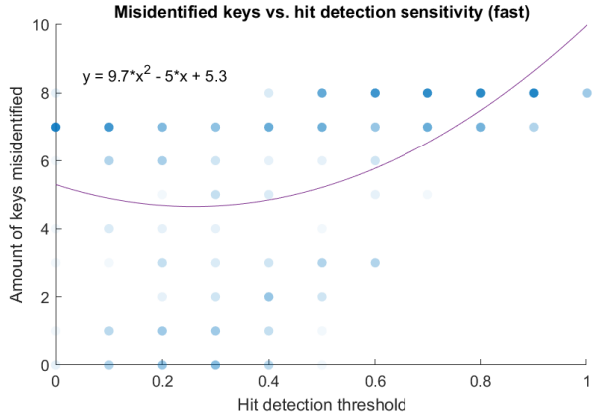

Figure 19: Scatter plot with quadratic trend line showing relation between misidentified keys and hit detection sensitivity, minimum of trend line is at x = 25/97, playing in fast manner

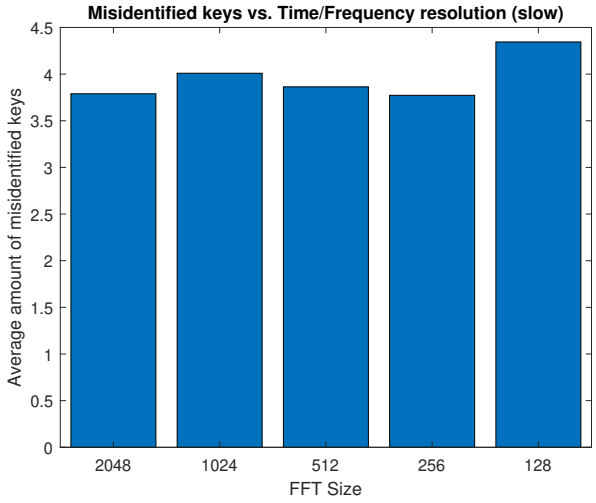

Figure 20: Bar plot showing average amount of misidentified keys per FFT window size, playing in slow manner

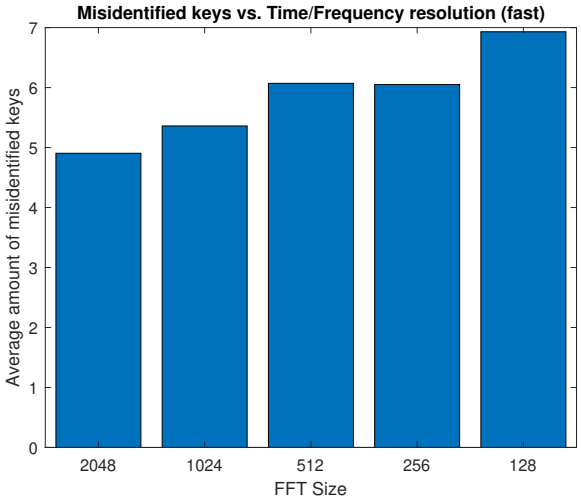

Figure 21: Bar plot showing average amount of misidentified keys per FFT window size, playing in fast manner

D. Similarity of Improvised Melody to Original

The results of the improvisation AI's experiment can be seen in figure 22, in which the lines represent three different melodies; Marry Had a Little Lamb consisting of 26 notes, Frere Jacques consisting of 32 notes, and Incy Wincy Spider consisting of 44 notes. These melodies were chosen because of their distinctive melodies and length. The modification factor is how much the unseen transitions probabilities were increased by and the similarity score is the number denoting how similar two pieces are, where a higher number indicates less similarity.

There is a general upward trend across all three lines, indicating that when the modification factor is increased, the improvised melody becomes less similar to original melody. Additionally, the lines are vertically ordered from high to low in reverse order of their length, suggesting that when the provided sequence is longer, the resulting similarity score is lower across all modification factors. This means that when

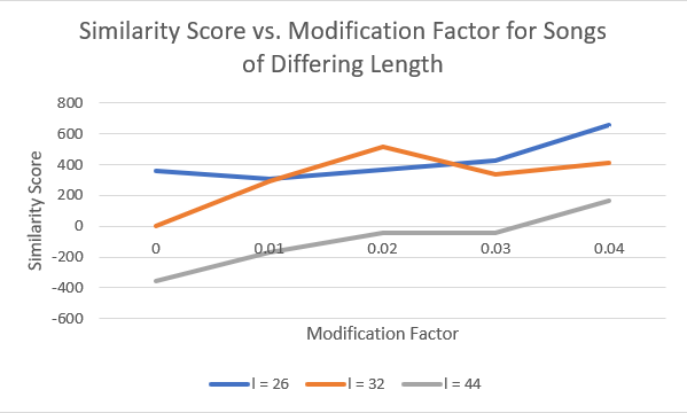

Figure 22: Results for the Improvisation Experiment

the provided sequence is longer, the improvised piece is more similar to it. Furthermore, the correlation coefficients are 0.82 for the shortest melody, 0.70 for the intermediate length melody, and 0.96 for the longest melody. It is important to note that the x-axis does not have any special significance here, the important part of the similarity score is the difference between the scores rather than the scores themselves.

VIII. Discussion

A. Triangle 1 and Triangle 2

From the results of the first experiment, found in Figure 16, it seems Triangle 2 performs better than Triangle 1. This could be explained by the more abrupt halt caused by the Triangle 1 path. By coming to a halt above the key before striking down, at times the wrong key was struck. Because, the setup shakes when a movement is halted abruptly. With a one sided T-test testing if Triangle 2 is significantly more accurate to Triangle 1 we found a p value of 0.0014. With an alpha value of 0.25 this is indeed significant.

B. Tempo and accuracy

Curiously, as seen in Figure 17, tempo two with half a second delay seems to make the least mistakes. Tempo six would be expected to have the least mistakes. When we test the two tempos for significance with a two sided T-test and an alpha of 0.5, we find a p-value of 0.44. So, the difference is significant in favor of tempo 6. Tempo 2 is not significantly better than 6, it is worse. However, at a delay of 0.5 it seems to be performing better, with no statistical data to back it.

C. Similarity of Improvised Sequences

The results shown in 22 agree with the prediction that similarity decreases as the probabilities of unseen transitions increases. This makes sense as increasing this probability encourages the robot to play notes which are less based on the input melody. Although it appears that the length of the input melody increases similarity to the improvised melody, it is not fair to draw this conclusion, as there are only three lines to compare and the lines overlap at certain points. All correlation coefficients suggest that these two factors are correlated. The lowest one of 0.70 can be explained by the spike in the middle, which may be the result of an unfair way of measuring melodic similarity. This may be the result of the similarity score being calculated without considering the timing of the hits, which may play a large role in their similarity. That being said, the way of calculating similarity seems acceptable seeing as the other two correlation coefficients are relatively high. Another potential shortcoming is that it may not be fair to assume the correlation is linear, so a correlation score may not even be appropriate. This again comes down to the difficulty of quantitatively measuring music.

Another interesting observation which can be made here relates the point at which the unseen transition probabilities become higher than the seen transition probabilities. This point occurs somewhere between the modification factor of 0.02 and 0.03 for each melody. This may be coincidence as there are only three melodies to base this assumption off of and further investigation is required.

D. Sound Processing

The sound processing works in simple cases, however with smaller window sizes and faster playing leading to overlapping notes as illustrated in Figure 23 and 24 sound processing breaks down. In Table I a significant difference in better performance with regards to keys identified is seen, while the results with the optimized parameters are significantly better (t-test = 1) as seen in Table II.

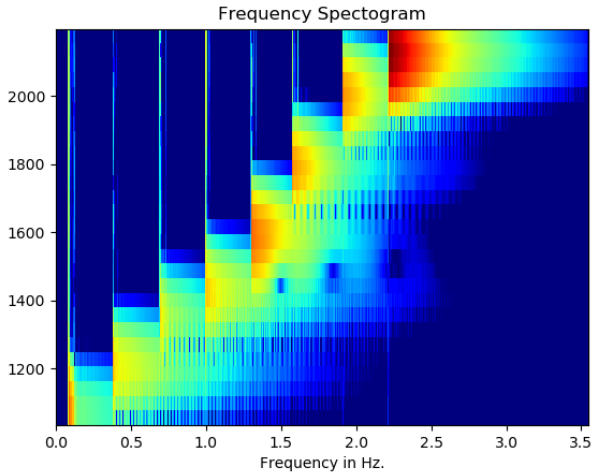

Figure 23: Spectogram of playing xylophone with FFT window size 512, playing in fast manner

IX. Conclusion

The goal of this paper is to optimize the configuration of the closed loop system of the ”xylobot”. Using computer vision, sound processing, control components, a simulation, and an artificial intelligence, the robotic xylophone player can play a tune given by a human player and continue improvising the tune using a Markov chain. Based on the improvisation AI’s experiment, it is safe to say that the higher the probability of unseen note combinations appearing, the less similar the

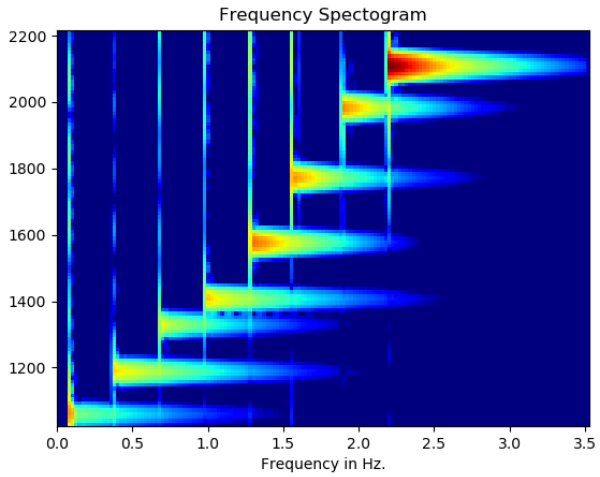

Figure 24: Spectogram of playing xylophone with FFT window size 2048, playing in fast manner

improvised sequence is to the original. However, the idea behind the experiment was to give the robot more freedom to improvise, and when the the two melodies are less similar, the robot can be considered more creative and less constrained. A trade-off must be made between keeping to the theme of the input melody and coming up with an original melody The chosen modification factor should reflect this, as a high modification factor allows more freedom and less similarity, and a low modification factor allows less freedom and more similarity. With the control experiments it was discovered that Triangle 2 is the best method for hitting notes at any pace. A slower pace is more accurate overall, but Tempo 2 seems to be working well, however, without statistical data to back it. More extensive research will have to be done to find out for certain. From the sound processing results it can be concluded that optimized parameters do significantly increase the accuracy in detection of keys. In the future it would be interesting to explore how accurate the timing of the sound processing is and how it would deal with noise and disturbances. If the precise characteristics of the robot could be known and how accurately it can hit a key at a given time when given a command, random sequences could be generated and played by the robot and then compared with the result of the sound processing. This could be done to further explore the impact of the FFT window size and time resolution on the performance in timing of a Xylobot-like setup and eliminates inconsistencies due to a human generating the samples. Noise could also be added to the signal in the form of white noise or the signal degraded to simulate worse microphones. This could be interesting when costs need to be reduced if a Xylobot-like robot would for example be produced as a toy for children.

References

[1] A. Taheri, A. Meghdari, M. Alemi, and H. Pouretemad, "Teaching music to children with autism: A social robotics challenge," *Scientia Iranica*, vol. 26, pp. 40–58, 02 2019.

[2] T. S. Perry, "Inside an experimental robotics class: A robot sketch artist, a robot that plays dominos, and more," 06 2016.

[3] "Graphical user interfaces with tk — python 3.8.1," https://docs.python.org/3/library/tk.html, (Accessed on 01/16/2020).

[4] A. Aristidou and J. Lasenby, "Fabrik: A fast, iterative solver for the inverse kinematics problem," *Graphical Models*, vol. 73, no. 5, pp. 243–260, 2011.

[5] E. J. Linskens and G. Schoenmakers, "Music improvisation using markov chains," 2014.

[6] H. Hazimeh and C. Zhai, "Axiomatic analysis of smoothing methods in language models for pseudo-relevance feedback," in *Proceedings of the 2015 International Conference on The Theory of Information Retrieval*, ser. ICTIR '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 141–150. [Online]. Available: https://doi.org/10.1145/2808194.2809471

[7] "Quantization," https://www.mediacollege.com/glossary/q/quantization.html, (Accessed on 01/22/2020).

[8] G. Bradski, "Opencv," https://opencv.org/, (Accessed on 01/22/2020).

[9] O. Huggins, "Opencv for detecting edges, lines and shapes," https://hub.packtpub.com/opencv-detecting-edges-lines-shapes/, (Accessed on 01/22/2020).

[10] "drawing a rectangle around a color as shown? - opencv q&a forum," https://answers.opencv.org/question/200861/drawing-a-rectangle-around-a-color-as-shown/, (Accessed on 01/22/2020).

[11] J. O. Smith III, *Spectral audio signal processing*. W3K publishing, 2011.

[12] J. W. Cooley, P. A. Lewis, and P. D. Welch, "Historical notes on the fast fourier transform," *Proceedings of the IEEE*, vol. 55, no. 10, pp. 1675–1677, 1967.

[13] W. T. Cochran, J. W. Cooley, D. L. Favin, H. D. Helms, R. A. Kaenel, W. W. Lang, G. C. Maling, D. E. Nelson, C. M. Rader, and P. D. Welch, "What is the fast fourier transform?" *Proceedings of the IEEE*, vol. 55, no. 10, pp. 1664–1674, 1967.

[14] cohortor.org, "Tuner - gstrings free - apps on google play," https://play.google.com/store/apps/details?id=org.cohortor.gstrings, (Accessed on 01/21/2020).

[15] Stonekick, "Pitched tuner," https://play.google.com/store/apps/details?id=com.stonekick.tuner, (Accessed on 01/21/2020).

[16] D. Gerhard, *Pitch extraction and fundamental frequency: History and current techniques*. Department of Computer Science, University of Regina Regina, Canada, 2003.

[17] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. Jarrod Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. Carey, İ. Polat, Y. Feng, E. W. Moore, J. Vand erPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and S. . . Contributors, "SciPy 1.0–Fundamental Algorithms for Scientific Computing in Python," *arXiv e-prints*, p. arXiv:1907.10121, Jul 2019.

[18] J. Griffié, L. Boelen, G. Burn, A. P. Cope, and D. M. Owen, "Topographic prominence as a method for cluster identification in single-molecule localisation data," *Journal of biophotonics*, vol. 8, no. 11-12, pp. 925–934, 2015.

[19] J. Blades, *Percussion instruments and their history*. Bold Strummer Limited, 1992.

[20] H. Pham, "Pyaudio: Portaudio v19 python bindings," *URL: https://people. csail. mit. edu/hubert/pyaudio*, 2006.

[21] D. Luo, W. Luo, R. Yang, and J. Huang, "Compression history identification for digital audio signal," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2012, pp. 1733–1736.

[22] R. N. Czerwinski and D. L. Jones, "Adaptive short-time fourier analysis," *IEEE Signal Processing Letters*, vol. 4, no. 2, pp. 42–45, 1997.

[23] P. Podder, T. Z. Khan, M. H. Khan, and M. M. Rahman, "Comparative performance analysis of hamming, hanning and blackman window," *International Journal of Computer Applications*, vol. 96, no. 18, 2014.

[24] P. Muralidhar, D. Nataraj, V. Lokesh Raju, and S. K. Naik, "Implementation of different fir high pass filters using fractional kaiser window," in *2010 2nd International Conference on Signal Processing Systems*, vol. 2. IEEE, 2010, pp. V2–651.

[25] MATLAB, "Short-time fft," [Accessed: 21-01-2020]. [Online]. Available: http://web.archive.org/web/20200121005359/https://nl.mathworks.com/help/dsp/ref/dsp.stft.html
[26] B. Logan, D. P. Ellis, and A. Berenzweig, "Toward evaluation techniques for music similarity," 2003.
[27] Z. Rafii and B. Pardo, "Music/voice separation using the similarity matrix." in *ISMIR*, 2012, pp. 583–588.
[28] "Triads - music theory academy," https://www.musictheoryacademy.com/understanding-music/triads/, (Accessed on 01/22/2020).

APPENDIX A
FUNDAMENTAL FREQUENCIES OF XYLOPHONE KEYS

Note	Frequency Xylophone (in Hz.)	Standard Tuning Frequency (in Hz.)
C_6	1052.9	1046.5
D_6	1181.8	1174.7
E_6	1324.5	1318.5
F_6	1399.4	1396.9
G_6	1570.1	1568.0
A_6	1767.1	1760.0
B_6	1975.2	1975.5
C_7	2102.2	2093.0

Table III: Fundamental Frequencies of Xylophone keys