Radboud University Nijmegen

Faculty of Social Sciences

Artificial Intelligence

M. Biondina

Bachelor Thesis

AI generated visual accompaniment for music

- Machine learning techniques for composing visual accompaniment for music shows -

Student:	Matthijs Biondina
Studies:	Artificial Intelligence
Semester:	2016-2017 semester 2
Student ID:	S4284046
Birth date:	14-10-1995
E-mail:	m.biondina@student.ru.nl

Supervisors: Peter Desain
Artificial Intelligence, Faculty of Social Sciences, Radboud University Nijmegen

Franc Grootjen
Artificial Intelligence, Faculty of Social Sciences, Radboud University Nijmegen

Nijmegen, 18-08-2017

ABSTRACT

In this study a method for artificially composing visual accompaniment for music pieces is proposed. We analyze whether the proposed method composes visual accompaniments that are comparable in quality to visual accompaniments made by a human artist. It was found that visual accompaniments composed by the proposed methods are judged significantly lower in quality than their human-made counterparts. Additionally, it was found that the performance of the proposed method did not differ significantly from a pseudo-random approach to composing visual accompaniments. Despite these results, this method might provide a framework for future research on this topic.

INTRODUCTION

Since the invention of music, people have been interested in creating accompaniments for this music in a visually enjoyable manner, for example in the form of dance. Recently, due to radical advancements in technology, the possibilities for creating visual accompaniments for music have increased enormously. Now we compose visual accompaniments for music in all sorts of manners; from speakers that contain small fountains which dance on the rhythm of the music, to massive light shows at concerts where powerful laser-beams and impressive flame throwers turn listening to music into an entirely different experience.

Research has been done on the topic of artificially generating enjoyable music. For example, De Manteras & Arcos (2002) give an in-depth analysis of different systems that researchers created for the artificial generation of music.

The earliest research in the field of artificially generated music was done by Hiller and Isaacson's (1959), who used a computer to compose a classical music composition named *"Illiac Suite"* (later renamed as "*String Quartet No. 4"*). They used a pseudorandom system to generate notes with Markov chains. Next, these notes were tested based on a number of heuristics. Notes that did not adhere to the heuristics were discarded. Additionally, when no notes were available that matched the heuristics, a backtracking process was initiated to avoid this situation.

Later, Rader (1974) designed an AI application for artificially generating music based on a rule-based approach. Rader separated the process of generating overall harmony and specific notes, however the methods he used for both categories were largely similar. Generation was based on a set of rules, that specified how notes and chords can be put together. On top of that, Rader used a set of "*applicability rules"* which specified which rules could be used in which situations. Whenever there was at least one *applicability rule* that specified that a certain rule does not fit in the music at a specific situation, then it could not be used. Lastly, Rader introduced a third set of rules, the "*weighting rules"*, which specified the probability that a certain rule could be used, based on weights assigned to the *applicability rules*. With this system, Rader managed to compose music that "sounds mediocre to the professional although usually pleasing to the layman."

However, not much research exists for generating visual accompaniment for music. In many cases creating such an accompaniment is a tedious and often time-consuming process. Therefore, it would be useful to explore possibilities in automating this process.

The problem of creating an interesting visual accompaniment for a piece of music can be divided into three subproblems. First of all, the visual effects of the accompaniment should match the rhythm and energy of the music. Secondly, the visual effects accompanying the music should fit well together, so that the visual accompaniment is perceived as a coherent whole, rather than random bursts of light, water, etc. on the beat of the music. Thirdly, the visual effects of the accompaniment should be varied enough to create a visually stimulating experience.

In this study, a technique for composing visual accompaniments for music pieces was developed that can be used on a wide variety of visual accompaniment systems. In this study, it was chosen to use a *launchpad* layout as visual accompaniment for the music. The launchpad layout provides a basic framework for creating light shows, can be simulated easily, and is small enough to run experiments without the need for a large setup.

A launchpad is an electronic music instrument that has gained popularity in recent years. The display consists of an 8x8 grid of illuminated, square buttons, surrounded by a number of additional buttons towards the edge of the instrument. The specific layout used was derived from the *Novation Launchpad Pro.* In this layout there are an additional eight illuminated, round buttons on all four sides of the grid (see image). A launchpad is commonly used for playing music. An artist assigns a sound clip to each button on the launchpad, which is played when the button is pressed. As long as the artist remembers which sound belongs to which button, the launchpad can be

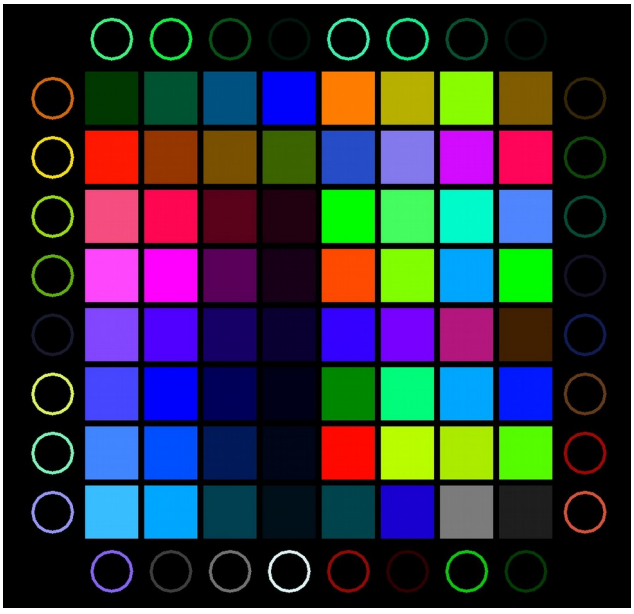

Figure 1: layout of the Novation Launchpad Pro

played like any key-based instrument. Additionally, the artist can assign light effects to certain buttons. Meaning that, when a button is pressed, several buttons on the display light up in a predefined color and sequence, creating interesting visual effects while playing. Alternatively, the launchpad can also be used for making light shows alone, which is the focus of this study. In that case, the artist predefines exactly which lights will light up during the course of an entire song. This process is very time consuming and it can take up to several hours to predefine a light show for a minute of a song. From the artist's perspective it may therefore be useful to have a program that performs the same job automatically.

The program proposed in this study could be used to compose any visual show accompanying music. In this study, the program was used solely for the purpose of composing launchpad light shows. However, the program could be used for any visual display, given that the following requirements are met: Firstly, a set of training data must be available to train the program. The program attempts to replicate behavior from human-made example shows and apply this behavior to the music of the new song. Therefore, the quality of the shows composed by the program is limited by the quality of the human-made examples, the number of examples provided, and the similarity between the music in the example shows and the music for the show that is to be composed. (e.g. if the program is only given examples of shows made for classical music, it will not perform well at making a show for a rock song). Secondly, it must be possible to create an abstract representation of the layout of the instrument used to produce the show that is consistent between the training data and the final light show. For example, the layout of the *Novation Launchpad Pro*, used in the study, consists of 96 illuminated buttons, which are numbered 28 to 123. This layout is consistent between all *Novation Launchpad Pros*, therefore any light show made on a *Novation Launchpad Pro* can be used as training data for the program, and a light show composed by the program can be played on any *Novation Launchpad Pro*. Similarly, if the program would be used to compose a water show or a firework show, it would be required that the human-made shows used for the training data are made either for exactly the same layout, or on a similar layout, where elements can be paired one-to-one with the new layout.

In order to artificially compose enjoyable light shows, one must know what it is that makes human-made light shows enjoyable. Unfortunately, no research has

been done on this topic. However if one views light shows as an extension of the music they are made for, then one might be able to apply the same heuristics that make music enjoyable to light shows.

Minsky (1981) compares listening to a piece of music to watching a scene in a room. Minsky says that in each person's mind work many *agents* with very specific tasks. For example one *agent* might recognize a few small scraps from the visual field, another *agent* might recognize a shape in these scraps, and another *agent* might recognize this shape as part of a piece of furniture. Similarly, when it comes to music, a person's mind might have many *agents* dedicated to recognizing different parts of music. They might have one agent solely dedicated to recognizing rhythm, another that is capable of recognizing simple melodies, and another that assigns meaning to the music on a much higher level. Within this system there exists a hierarchy of layers, where each layer processes a more abstract and “meaning” oriented version of the information received from the layer below. As Minsky puts it: “Relations at each level, turn to Thing at next above; more easily remembered and compared.” This means that, while listening to music, the mind does not only process “the now”, but also searches and remembers meaning over a broader spectrum of time. As a result, when a rhythm is more monotonic or a melody more simplistic, the agents higher up in the hierarchy become less excited. De Manteras & Arcos (2002) state that a neuron's firing rate decreases over time when the neuron repeatedly receives the same input. This effect is called *habituation*. This explains that music is perceived as more interesting when it contains a certain amount of variation, “that is, when it contains alterations in dynamic, pitch, and rhythm.” Based on this, one can conclude that the light shows should also contain sufficient alterations in dynamic, “pitch,” and rhythm. Since the rhythm of the light show should match the rhythm , pitch and loudness of the music, these heuristics should be covered as long as the composed light show sufficiently fits the music. However, to prevent habituation, it is necessary to place restrictions on the program that force it to include a sufficient variety of visual effects in its light shows; hence creating light shows that are enjoyable to watch.

One might also wonder whether the light effects used have any meaning for the audience that goes beyond having matching rhythms. Bolivar et al. (1994) found that people are able to “assess the degree of audiovisual semantic congruency” between

video clips and accompanying music. In their study, they showed participants aggressive/friendly video clips accompanied by aggressive/friendly music, and asked the participants to judge whether the music matched the clip. Although the comparison between video clips and light shows is somewhat abstract, this does show that people are able to perceive incongruence between visual and auditory stimuli. This suggests that the application needs to be capable of depicting the energy of the songs to a level beyond merely matching the rhythm.

Many artificial music generation systems depend on a specified set of rules to compose this music. Since no research has been done before regarding a general approach to visually accompanying music, it would be difficult to formulate such rules for our purpose. Alternatively, some artificial music generation programs work by iteratively improving a composition throughout a number of generations with a genetic algorithm (Biles, 1994). The fitness function in these systems is often implemented algorithmically. However, Wiggins (1998) points out that there exists no general formalized fitness function for judging the quality of music. Therefore it is often necessary to let a human operator subjectively judge the quality of a piece of music generated by the system. In this case we speak of an *Interactive Genetic Algorithm* For the system proposed in this study, neither of these methods is optimal, as there are no general heuristics for judging the quality of a light show, making it difficult to define a rule-based system; and the premise of this study is to create an automated system, therefore making the system interactive would contradict with our interests.

METHODS

Due to the impracticalities of designing the system as a rule-based AI or making the system interactive, instead a Machine Learning approach was chosen, where the program learns from example light shows made by a human. The program then replicates the behaviors of the human-made light show when presented with a new song.

Training Data

We contacted Youtuber *InspirAspir*, who was willing to provide us with several light shows he made for his Youtube channel, and gave us permission to use these light shows for training and testing purposes. In total, seven of his light shows have been used, namely:

Table 1 (list of light shows used, made by InspirAspir)

Song	**Artist(s)**
Abyss	Kaskobi
Blow Up	ViperActive
Invincible	DEAF KEV
Lost Woods (remix)	InspirAspir
Roses	The Chainsmokers
Wizards in Winter	Trans-Siberian Orchestra

These songs provided a reasonably wide variety of genres, ranging from dubstep to alternative rock.

The data files InspirAspir provided were formatted in the form of *Ableton Live* project files. These files contain several thousand lines of commands, specifying at which time during the show a certain light should light up, for how long, and in what color.

Since the launchpad itself can not decode these files itself, but instead receives MIDI events from an intermediate program (like *Ableton Live*), these files had to be converted to MIDI files. Since MIDI files are sorted in chronological order, whereas the project files are sorted on button number, a buffer had to be implemented in the program that stored all *MidiNoteEvents* from the project files, sorted them on time, and wrote these to a MIDI file afterwards. The MIDI code needed to light up one button on the launchpad can be easily derived from the corresponding *MidiNoteEvent* in the project file.

```
<MidiNoteEvent Time="1.3798076923076923" Duration="0.173076923076923073" Velocity="33.9999924" OffVelocity="64" IsEnabled="true" />
<MidiNoteEvent Time="1.5528846153846154" Duration="0.0865384615384615363" Velocity="34.9999924" OffVelocity="64" IsEnabled="true" />
<MidiNoteEvent Time="1.6394230769230769" Duration="0.0865384615384615363" Velocity="35.9999924" OffVelocity="64" IsEnabled="true" />
<MidiNoteEvent Time="12.125" Duration="0.0625" Velocity="49.0000114" OffVelocity="64" IsEnabled="true" />
<MidiNoteEvent Time="13.379807692307692" Duration="0.173076923076923073" Velocity="49.0000076" OffVelocity="64" IsEnabled="true" />
<MidiNoteEvent Time="13.552884615384615" Duration="0.0865384615384615363" Velocity="50.0000076" OffVelocity="64" IsEnabled="true" />
<MidiNoteEvent Time="13.639423076923077" Duration="0.0865384615384615363" Velocity="51.0000076" OffVelocity="64" IsEnabled="true" />
<MidiNoteEvent Time="19" Duration="0.0625" Velocity="33.0000038" OffVelocity="64" IsEnabled="true" />
<MidiNoteEvent Time="19.0625" Duration="0.0625" Velocity="34.0000038" OffVelocity="64" IsEnabled="true" />
<MidiNoteEvent Time="19.125" Duration="0.0625" Velocity="35.0000038" OffVelocity="64" IsEnabled="true" />
<MidiNoteEvent Time="21.25" Duration="0.0625" Velocity="21" OffVelocity="64" IsEnabled="true" />
<MidiNoteEvent Time="21.3125" Duration="0.0625" Velocity="22.0000019" OffVelocity="64" IsEnabled="true" />
<MidiNoteEvent Time="21.375" Duration="0.0625" Velocity="23.0000019" OffVelocity="64" IsEnabled="true" />
<MidiNoteEvent Time="23.314935064935064" Duration="0.0625" Velocity="49.0000076" OffVelocity="64" IsEnabled="true" />
<MidiNoteEvent Time="23.377435064935064" Duration="0.0625" Velocity="50.0000076" OffVelocity="64" IsEnabled="true" />
<MidiNoteEvent Time="23.439935064935064" Duration="0.0625" Velocity="51.0000076" OffVelocity="64" IsEnabled="true" />
<MidiNoteEvent Time="25" Duration="0.0625" Velocity="3" OffVelocity="64" IsEnabled="true" />
<MidiNoteEvent Time="25.0625" Duration="0.0625" Velocity="2.99999976" OffVelocity="64" IsEnabled="true" />
<MidiNoteEvent Time="25.125" Duration="0.0625" Velocity="1" OffVelocity="64" IsEnabled="true" />
<MidiNoteEvent Time="29.25" Duration="0.0625" Velocity="21" OffVelocity="64" IsEnabled="true" />
<MidiNoteEvent Time="29.3125" Duration="0.0625" Velocity="22.0000019" OffVelocity="64" IsEnabled="true" />
<MidiNoteEvent Time="29.375" Duration="0.0625" Velocity="23.0000019" OffVelocity="64" IsEnabled="true" />
<MidiNoteEvent Time="31.25" Duration="0.0625" Velocity="49.0000076" OffVelocity="64" IsEnabled="true" />
<MidiNoteEvent Time="31.3125" Duration="0.0625" Velocity="50.0000076" OffVelocity="64" IsEnabled="true" />
<MidiNoteEvent Time="31.375" Duration="0.0625" Velocity="51.0000076" OffVelocity="64" IsEnabled="true" />
<MidiNoteEvent Time="32.125" Duration="0.0625" Velocity="33.9999924" OffVelocity="64" IsEnabled="true" />
<MidiNoteEvent Time="32.25" Duration="0.0625" Velocity="33.9999924" OffVelocity="64" IsEnabled="true" />
```

Figure 2: several lines of code from an Ableton Live project file. Each line encodes a flash of a single light on the launchpad (an event). The tags that are relevant for translating an event to MIDI code are as follows:

- *Time: the amount of time from the beginning of the show to the start of the event*
- *Duration: the amount of time that the light stays lit*
- *Velocity: an integer value describing with which color the light flashes from a list of predefined colors*

For a detailed description on how light shows are encoded in MIDI, see Appendix I.

Simulating the Launchpad

Since Radboud University does not own a launchpad, a program had to be written to simulate the light shows on a computer. As well as bypassing any technical issues that could have arisen with integrating the program with a physical launchpad device, this made it possible to neatly integrate the light shows with the experimental setup later on. The layout of the *Novation Launchpad Pro* used by InspirAspir was replicated and placed over a black background to maximize the amount of contrast between the lights and the background. The numbers assigned to each button were copied from the *Novation Launchpad Pro* (see figure 3). This took some experimentation and manual adjustment, since no ready-made scheme existed for this layout. Within the program, the layout can be easily configured and changed to any layout desired.

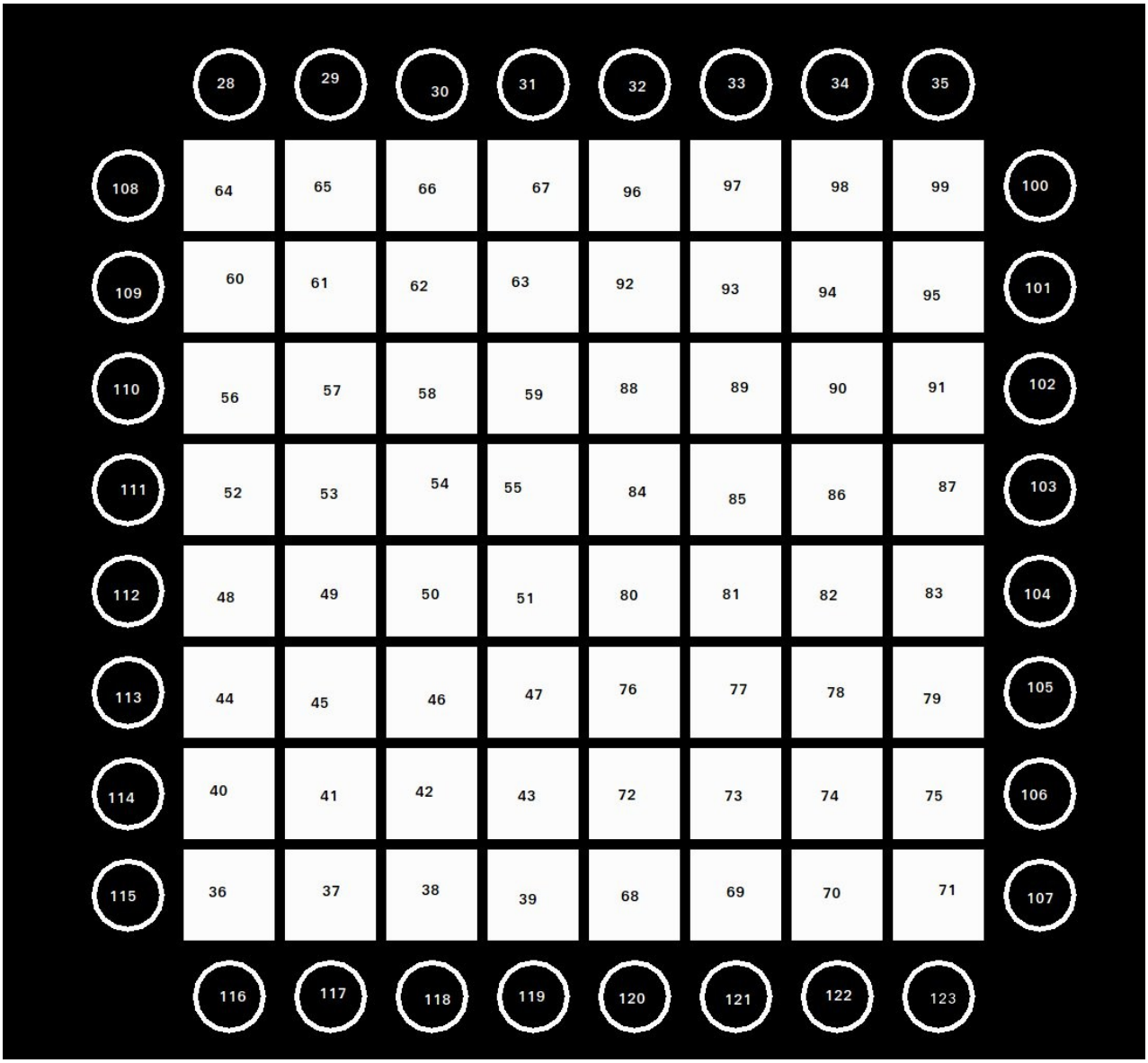

Figure 3: the layout of the simulation display, as well as the numbers assigned to each light

The launchpad is able to display 128 predefined colors. The number assigned to these colors were also copied from the *Novation Launchpad Pro* (see figure 4). By doing so, the simulations of the light shows in the data set matched as closely to the originals made by InspirAspir as possible.

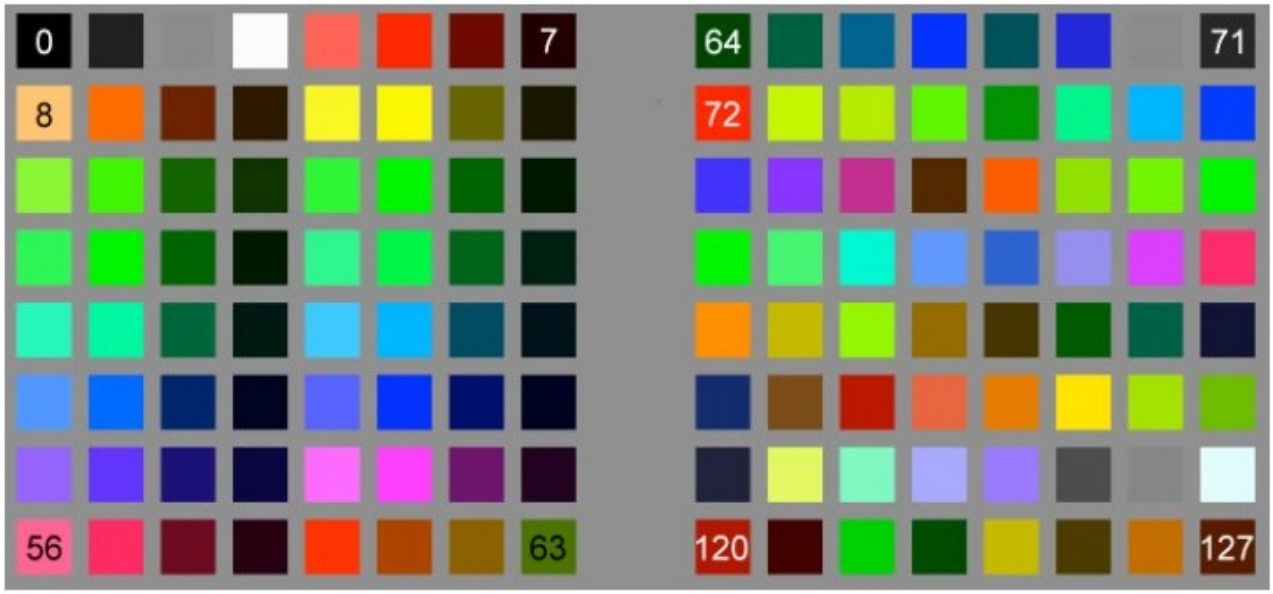

Figure 4: numbering of the 128 colors displayed by the launchpad. Source: LAUNCHPAD PRO Programmer's Reference Guide

While simulating a light show, the program displays the light show and plays the accompanying song in parallel. The program keeps an internal buffer of the MIDI file describing the light show and executes these hex-bytes one by one. As long as a MIDI event is preceded by a delay of 0 ms, the program updates the internal state of the simulation, but does not update the visual display. Once the program finds a MIDI event with a delay larger than 0 ms, it updates the display and pauses the program for the specified amount of time before continuing the process of reading and updating the simulation. The simulation maintains an internal clock, which is used as reference when determining how long the program needs to wait given a certain delay. This guarantees that the timing of the light show is not thrown off by any computation time needed to update the simulation. Otherwise this computation time might add up over the course of the song and create inconsistency between the timing of the displayed light show and the original MIDI file. The behavior of the simulation is illustrated in figure 5.

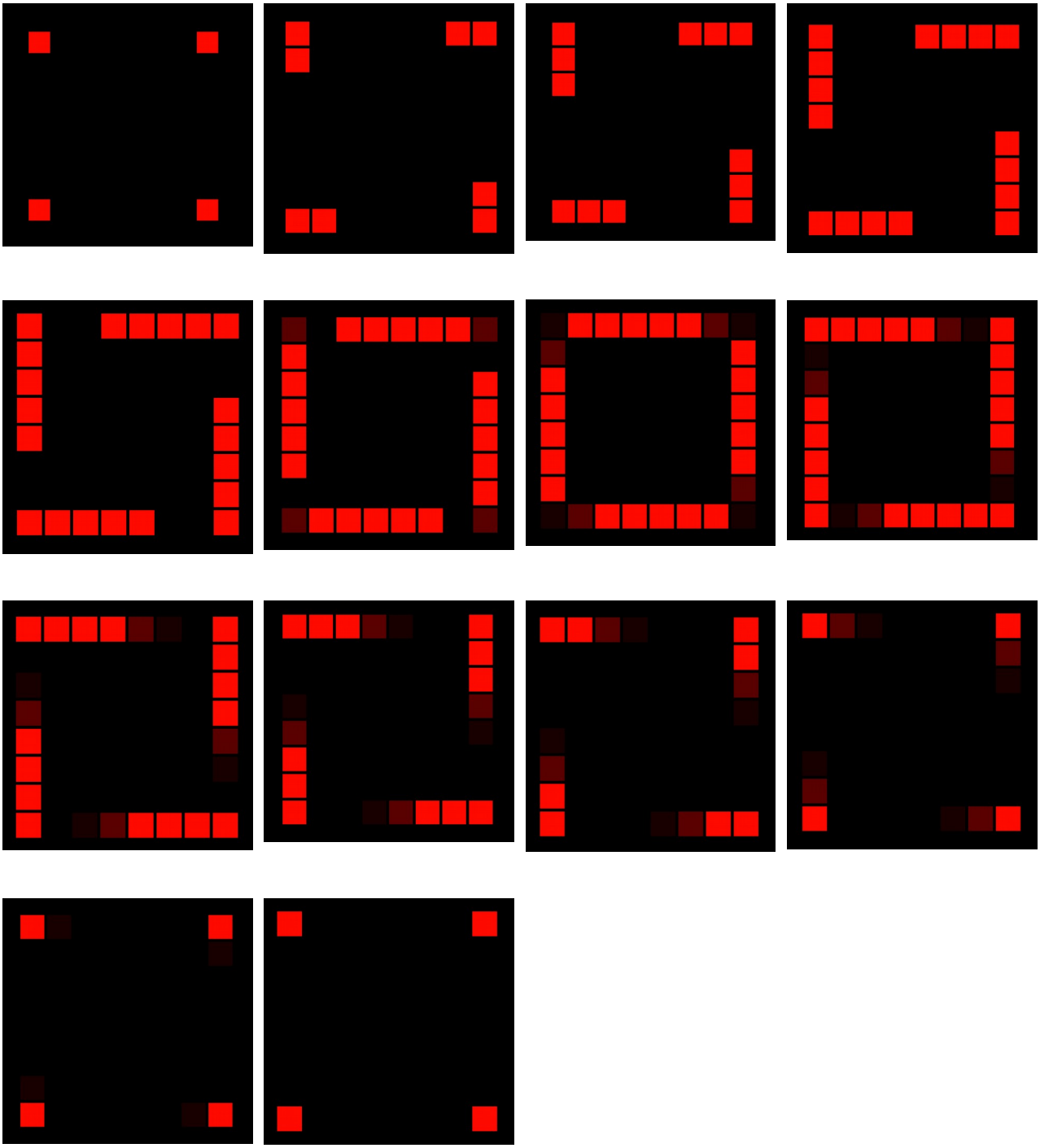

Figure 5: a sequence of changes to the visual display of the simulation. NB: light shows are generally fast paced; the sequence shown would only last circa half a second in real time.

Training Data

In order to replicate human behavior, the program needs to be able to pick and choose from a wide variety of options of visual effects from the original light shows. To this purpose, the original light shows were split up in separate effects. An effect is

defined as an amount of MIDI code which, through a sequence of light-off and light-on events, delays and updates, transforms the launchpad display from one state into another. An effect can only be played during a light show if the pattern of lights that are on on the display matches the pattern of lights of the starting state of the effect. Since the MIDI code of an effect only tells the launchpad which lights to turn on and off and in which color, a selected effect might otherwise leave lights on where they are supposed to be off and vice versa. Thus altering the visual appearance of the effect from its intended use. Effects are usually short, repeating sequences, lasting 0.5 to 2 seconds. For example, the sequence shown in (figure 5) would be stored as a single effect in the database. Occasionally, the light shows contained sequences that had no clear or repeating pattern. In those cases, we split up the sequences in separate effects at arbitrary points based on patterns that commonly occurred between other effects (see figure 6).

The corresponding MIDI code of these effects was stored as well as the context in which the effect was originally used, in the form of the music segment from the song over which the effect was displayed, the state of the visual display before the effect was played, and the color of the lights that were on before the effect was played. Combining these characteristics allows the algorithm to select effects that match the music and fit in well with the flow of the light show; both positionally and color wise. Segmentation of the light shows was done manually. An application was built that made it possible to manually play the light shows from the training data frame by frame, and create a cut, wherever a switch between two effects is made. This process was time-consuming. Once in the database , however, the separated effects can be used permanently for composing new shows. Certain patterns on the launchpad appear to occur frequently in transitions between effects (see figure 6).

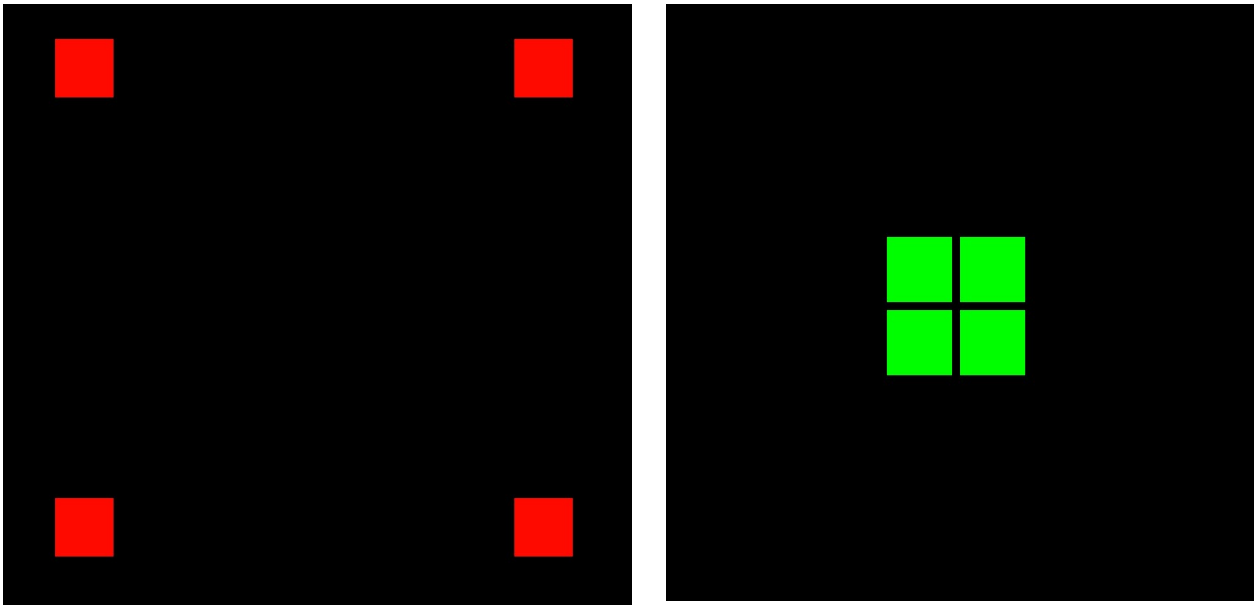

Figure 6: examples of patterns on the launchpad that often occur between effects.

Based on these characteristics, it should be possible to make an automated system for segmenting effects in a light show. This, combined with other heuristics like the average length of effects could be used to design an algorithm that performs this process artificially, for example with an artificial neural network. However, since time was limited and there was enough data to move onto artificially composing light shows, this possibility was not explored further. The training data consisted of roughly 1700 effects.

Composing Light Shows

The problem of composing new light shows was divided into three sub-problems: matching the emotion of the music and the light show, matching the rhythm of the music and the light show, and creating an optimal flow within the light show.

Emotion

In order to match the energy of the light show with the energy of the music, the program calculates similarity scores between a short clip from the song and the music clips corresponding to the effects in the database. The effects used in the original light shows convey the emotion of the song they were originally made for. For example, if part of a song conveys an uplifting message, then the effects originally placed over

this part of the song would also convey this uplifting message in some form or another. Hence, if a music clip from the new song is similar to a fragment of the original song, then the effect played over that fragment of the original song should also fit well with the fragment of the new song. Since artificially analyzing similarity between fragments of two songs on such an abstract level is both subjective and difficult to achieve, it was decided to compare music fragments with a more low-level approach. The similarity between two clips was determined by calculating the sum of squared errors between the spectrograms of both clips, divided by the length of the clips. Dividing by the length of the clip removes the program's bias towards shorter effects. In order to account for variations in rhythm between the two songs, the final similarity score is defined as the best fit between the two spectrograms given a delay before the new effect ranging from 0 to 250 ms. The final formula to calculate the similarity between a new clip (c) and a clip from a song (s) starting at (t_{start}) is as follows (see figure 7 for an illustration of this formula):

$$similarity(c, s, t_{start}) = \max_{delay=0}^{250} \frac{length(c)}{length(c) + \sum_{t=0}^{length(c)} (c(t) - s(t_{start} + delay + t))^2}$$

Although similarity is approximated on a more concrete level, the effect of this approach should be roughly the same. When two music clips convey the same sort of energy, the spectrograms of the two clips should also correlate more strongly, based on for example the volume and the direction of the pitch.

Rhythm

The option to compare the *bpm* (Beats Per Minute) of the new song to the *bpms* of the original songs was considered, however it has proven very difficult to artificially determine the *bpm* of a song. The algorithms delivered very inaccurate results and only returned a plausible number for about fifty percent of the songs. Because of this, it was decided to omit this possibility. Instead, in order to match the rhythm of the light show with the rhythm the same formula as mentioned above was used, with the difference that at the argmax is used rather than the maximum value. Since the effects are so short that generally they cover only a single beat in the music, it has no purpose to consider the overall bpm of the songs. The final formula to

determine the optimum amount of delay needed to insert a new effect with music clip (c) into a song (s) starting at (t_{start}) is as follows (see figure 7 for an illustration of this formula):

$$bestDelay(c, s, t_{start}) = \underset{delay=0}{\overset{250}{argmax}} \frac{length(c)}{length(c) + \sum_{t=0}^{length(c)} (c(t) - s(t_{start} + delay + t))^2}$$

By using the offset needed to create the best fit between two clips, the beats of both clips are lined up. Since the volume of a music clip is higher on the beat than off it, the best fit between two similar music clips should be at or near the point where the two beats align.

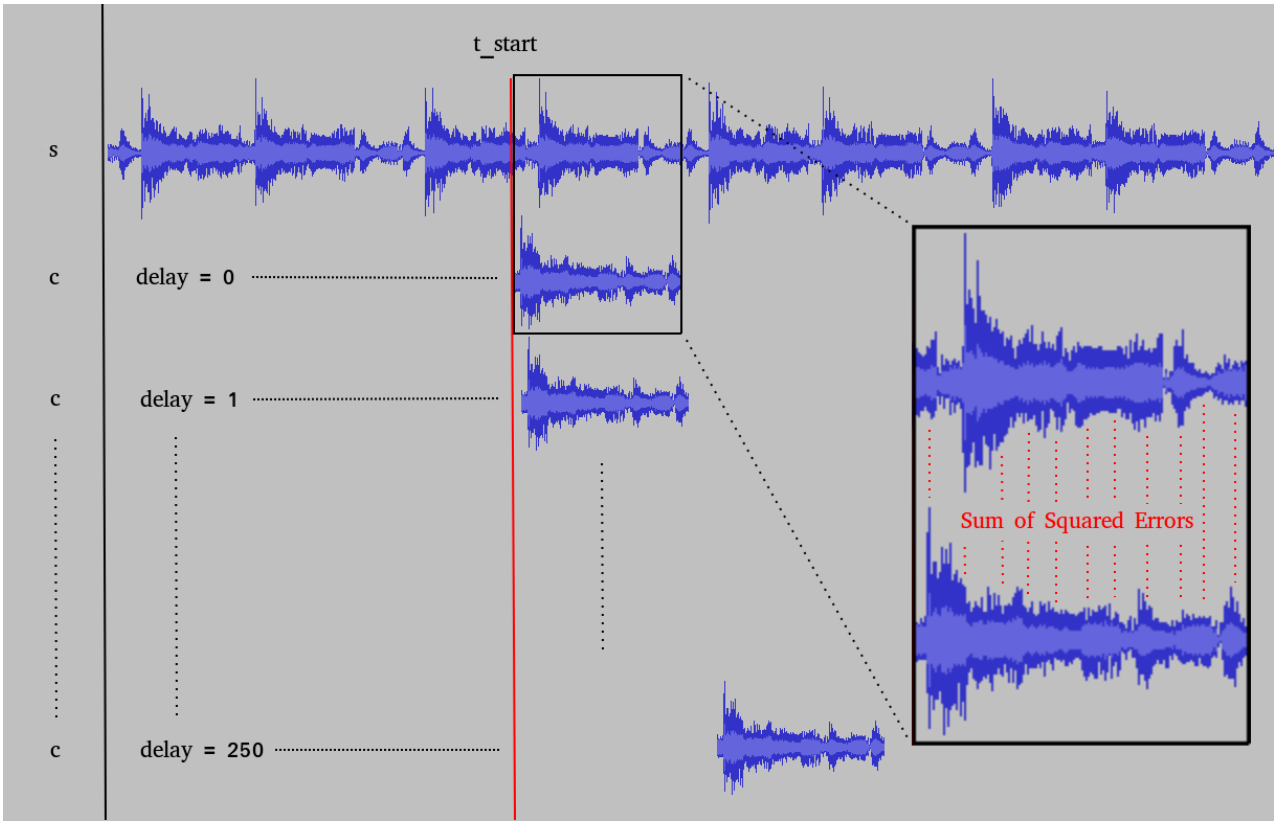

Figure 7: illustration of the similarity and bestDelay functions. For all possible delays between 0 and 250 milliseconds, the functions divide the length of the clip (c) by the sum of squared errors between (c) and the song (s) starting from (t_start + delay) over the length of (c). The similarity function returns the maximum value of this calculation and the bestDelay function returns the argmax.

Flow

The problem of guaranteeing good flow within the light show was divided in two. Firstly, the light show must not jump around positionally. What this means is that an effect must start from the same state on the launchpad that the previous effect has ended on. This was done by putting restrictions on the effects that the algorithm could choose from the database. While composing a new light show, the algorithm maintains an internal simulation of the launchpad show, registering which lights are on, and which color they have. When adding a new effect to the light show, the algorithm limits the effects it considers to effects that start in the same position as the simulation has at that time. Secondly, the colors displayed in the new light show should flow together nicely. In the original light shows in the training data, the starting color of an effect was generally similar to the color of the previous effect. However, since our algorithm can pick and choose effects from different songs and from different moments within the same song, there is no longer the guarantee that the colors of two effects placed together by the algorithm will also flow together nicely. In order to solve this, it was recorded how much the RGB values of the effects in the database changed after their first update from the state the previous effect ended in. Any time the algorithm adds a new effect to the light show, it looks at the colors of the lights that are turned on on the launchpad at that time and at the amount the colors of the new effect changed in the context of their original light show. Based on these values, the algorithm adjusts the colors of the new effect such that the change in colors with respect to the previous state of the launchpad is equal to the change in colors of the unadjusted effect with respect to the preceding state of the launchpad in its original light show. For example, if in the original light show an effect ended with all lights that are turned on being red, and after the first update of the next effect all lights that are turned on are still red (ie. there is no change in color in the original light show), then when the algorithm chooses to implement this effect in the light show and at that time all lights on the launchpad that are turned on are green, then the algorithm will adapt the colors of the effect in the new light show to also be green (ie. there is no change in color in the new light show).

Gradual Improvement

Estimating the similarity between to music clips is a rather time-consuming

process. Therefore, it is unfortunately not possible to perform an exhaustive search on the database. Instead a Greedy Best-First Search algorithm was implemented which uses localized Depth-First Search, inspired on the algorithm described by Hiller (1959). The algorithm composes a light show for a given song using the Greedy Best-First Search algorithm. At each step during the process the algorithm estimates the similarity between the next segment of the new song with the music clips from the effects in the database that have the same starting state as the internal simulation has at that time. Additionally, in order to guarantee an enjoyable level of variety in the light show, the algorithm is not allowed to use the same effect twice within a certain amount of time (2 minutes).The algorithm picks the effect with the highest similarity score and adds that to the light show. This process is repeated until a light show has been composed for the entire song. However a threshold variable is maintained during the process. At any point during the process, when the similarity score of the best-fitting effect does not exceed the threshold, the algorithm backtracks one step in the light show it has composed so far, and blacklists the last effect added to the light show. It then proceeds with searching for the best-fitting effect from that point on (excluding the ones that are blacklisted). If no effects are available, or the similarity scores of the available effects do not exceed the threshold, it backtracks another step, etc. During the first epoch, the threshold of the algorithm is set to 0. Hence, during the first epoch, the algorithm performs an ordinary Greedy Best-First Search. At the end of each epoch, the threshold value is increased to the similarity score of the worst-fitting effect in the light show. The light show it has composed during that epoch is written to a file, and the process starts over. Since the threshold value increases after each epoch, each next epoch will compose a light show that is slightly better than the previous. The program can be interrupted at any time, in which case the program will return the best light show so far.

Composing the Light Shows

For the experiment seven different light shows were composed by the program for seven different songs. Each of these seven songs was selected based on their similarity in style and genre to one of the songs from the initial data set. Six out of seven songs were made by the same artist that made the corresponding song from the initial data set. For one of the songs, *InspirAspir – Lost Woods (remix)* (a dubsteb

remix of a soundtrack song from the *Zelda* game franchise), no song made by the same artist existed that was similar in style. Instead a dubstep remix of another song from the *Zelda* soundtrack was selected made by a different artist, *Bitonal Landscape – Legend of Zelda: Main Theme (remix)*. The full list of songs used in the experiment is as follows:

Table 2 (the full list of songs used in the experiment)

InspirAspir	**AI**
Kaskobi – Abyss	*Kaskobi – Phantom*
ViperActive – Blow Up	*ViperActive – Atomic*
Bomb Squad – Damn Daniel	*Bomb Squad – Morphine*
DEAF KEV – Invincible	*DEAF KEV – Samurai*
InspirAspir – Lost Woods (remix)	*Bitonal Landscape – Legend of Zelda: Main Theme (remix)*
The Chainsmokers – Roses	*The Chainsmokers – Closer*
Trans-Siberian Orchestra – Wizards in Winter	*Trans-Siberian Orchestra – Wish Liszt*

All light shows have been trained for 5 hours, after which the last finished light show made by the program has been used during the experiment. During those 5 hours the program has managed to progress through fifteen to forty epochs, depending on the length of the song. Composing the light shows took a total of 35 hours. It must be noted that the program runs fully automatically, and does not require any form of human interaction.

Control Group

For each light show composed by the AI a pseudo-random variant was also made for the control group. In the control group, the algorithm selected random effects from all effects in the database who's starting state matched the state on the launchpad, without calculating the similarity between music clips of the new song and the original songs. A random amount of delay (between 0 and 250 milliseconds) was added before each effect, and no changes were made to the colors of the effects.

The Experiment

During the experiment a total of 34 participants have been asked to judge the quality of two light shows. The majority of the participants were students at Radboud University Nijmegen. Before the experiment participants have been asked to sign a letter of consent, to indicate that they consented to taking part in our study and did so out of free will. Additionally, participants were asked to explicitly declare that they did not have epilepsy, that they did not suffer from photosensitive epileptic seizures and that they did not have any other afflictions that could be triggered by visual cues.

The participants were instructed that they were going to watch two light shows made by Youtuber InspirAspir, and that they would be asked to answer several questions about these light shows to measure how they judged the quality of both light shows. The participants were assigned to the experimental group or the control group at random. Participants in the experimental group were shown one light show made by InspirAspir and the corresponding light show made by the AI program. Participants in the control group were shown one light show by InspirAspir and the corresponding light show made by the pseudo-random algorithm. The order in which participants watched the two light shows was randomized. In the end, the experimental group contained 20 participants and the control group contained 14 participants.

The decision to split the participants in two groups was made based on practical reasons. Participation in the experiment required circa 15 minutes of time. This was short enough that we could recruit volunteers from students who were spending time between lectures. Had a within-subject design been used, it would have been necessary to show participants three light shows each. In that case, participation would have required circa 20 to 25 minutes, which would have made it much more difficult to find the same number of participants.

The experiment has been performed with a double-blind design. At the start of the experiment each participant has been asked to write down a number on their questionnaire. This number could later be traced back to the condition they were in. The researcher did not see which number the participants wrote down, nor did the researcher see which light shows the participant was watching.

After watching each light show, the participant has been instructed to judge the

quality of the light show by indicating how much they agreed with nine statements about the light show. For each statement they could indicate that they fully disagreed, disagreed, had a neutral opinion, agreed or fully agreed with the statement. Their answers were scored on a scale from 1 to 5, where “fully disagree” corresponded with a score of 1, and “fully agree” corresponded with a score of 5. Questions 5 and 8 (see below) have been scored in reverse where “fully disagree” corresponded with a score of 5, and “fully agree” corresponded with a score of 1. The nine statements were as follows:

Table 3 (list of statements on the questionnaire)

Number	**Statement**
1	The effects matched well with the song at any given moment.
2	The effects flowed together nicely.
3	The effects paired well together.
4	The rhythm of the light show paired well with the rhythm of the song.
5	The light show was chaotic.
6	Overall, the light show conveyed the energy of the song well.
7	I enjoyed watching this light show.
8	There were moments during the light show when I was bored.
9	There were moments during the light show when I was amazed.

After watching both light shows and answering the questions corresponding to that light show, participants were informed that only one of the light shows they watched had been made by InspirAspir, whereas the other one had been made by an Artificial Intelligence Computer program. They were then asked which of the two they deemed was most likely made by the Artificial Intelligence computer program.

RESULTS

The results from the experiment show that the participants preferred the human-made light shows over both the light shows made by the AI approach and the pseudo-random approach. Both results are statistically significant ($p < 0.0005$ and $p < 0.01$).

Next it was analyzed whether the participants preferred the AI made light shows over the pseudo-randomly generated light shows. For this purpose, the gain-scores ($score_{computer} - score_{human}$) have been calculated for each of the nine statements. Next the gain-scores over all nine statements have been averaged to find the difference in overall perceived quality between the computer-made light shows and the human-made ones. There is no statistically significant effect between the gain-scores of the AI made light shows and the pseudo-randomly generated ones ($p >= 0.25$). Meaning that the participants did not prefer the AI approach over the pseudo-random approach.

$mean_{exp}$	$mean_{control}$	std_{exp}	$std_{control}$	n_{exp}	$n_{control}$	Cohens d	p
-0.7534	-0.7302	0.8425	0.4964	20	14	-0.0323	p >= 0.25

In fact, the gain-scores in the control group are slightly higher than the gain-scores in the experimental group, however this effect is nowhere near statistically significant either ($p >= 0.25$).

$mean_{control}$	$mean_{exp}$	$std_{control}$	std_{exp}	$n_{control}$	n_{exp}	Cohens d	p
-0.7302	-0.7534	0.4964	0.8425	14	20	0.0323	p >= 0.25

Next it has been analyzed how participants judged the quality of the computer-generated light shows with respect to the human-made ones on the different statements separately. Here some more interesting results have been found that might suggest that participants judge the quality of several aspects of the AI and pseudo-random approach differently, however again no statistically significant results are found. The gain-scores are slightly higher in the experimental group for statements 2, 3, 4, 5 and 6. However these results are not statistically significant ($p >= 0.25$ for statements 2, 3, 4 and 6 and $p < 0.20$ for statement 5). The gain-scores are slightly

higher in the control group for statements 1, 7, 8 and 9. However these results are not statistically significant either (p >= 0.25 for statements 1, 7 and 8 and p < 0.10 for statement 9).

Experimental condition was preferred over control condition:

Table 4

Statement*	**$mean_{exp}$**	**$mean_{control}$**	**std_{exp}**	**$std_{control}$**	**n_{exp}**	**$n_{control}$**	**Cohens d**	**p**
2	-0.6000	-0.7857	1.0954	1.1217	20	14	0.1679	p >= 0.25
3	-0.6316	-0.6429	1.1161	0.9288	19**	14	0.0108	p >= 0.25
4	-1.1500	-1.2857	1.3485	1.2666	20	14	0.1031	p >= 0.25
5	-0.5500	-1.0000	1.5720	0.9608	20	14	0.3315	p < 0.2
6	-0.6316	-1.0000	1.065	1.3587	19**	14	0.3078	p >= 0.25

** see table 3 for reference*

*** one participant did not answer this question for the AI approach*

Control condition was preferred over experimental condition:

Table 5

Statement*	**$mean_{control}$**	**$mean_{exp}$**	**$std_{control}$**	**std_{exp}**	**$n_{control}$**	**n_{exp}**	**Cohens d**	**p**
1	-0.9286	-1.1500	0.7300	1.3485	14	20	0.1135	p >= 0.25
7	-0.2143	-0.4000	1.1883	1.0954	14	20	0.1638	p >= 0.25
8	-0.5000	-0.8500	1.0190	1.2258	14	20	0.3053	p >= 0.25
9	-0.2143	-0.8000	1.1883	0.9515	14	20	0.5556	p < 0.1

** see table 3 for reference*

Lastly, it has been analyzed whether the participants were able to distinguish which of the two light shows they watched was made by a computer. In the experimental condition 45.00% of the participants correctly assessed which light show was made by a computer. In the control condition 42.86% of the participants correctly assessed which light show was made by a computer. The difference between these groups is not significant (p >= 0.25).

DISCUSSION

In this research it has been studied whether people would judge the quality of AI composed light shows lower than light shows composed by a human artist. Additionally it has been studied how participants judged the quality of AI composed light shows in comparison to pseudo-randomly generated light shows. Lastly it has been researched whether people would be able to correctly differentiate between a light show made by a human artist and a light show made by a computer.

To answer the first question, a t-test for paired observations has been done based on answers participants gave to nine statements regarding the quality of an AI composed light show and a human-made light show. Based on the results, one can conclude that people judge the quality the AI composed light shows as lower than the human-made light shows. The program proposed in this paper will therefore need to be improved before it can compose light shows on the same level as a human artist.

To answer the second question, a t-test for paired observations has been done based on the difference scores (gain-scores) of the human-made light shows and the AI-composed ones and the difference scores (gain-scores) of the human-made light shows and the pseudo-randomly generated ones. It has been found that there is no statistically significant difference between the gain-scores of the AI composed light shows and the pseudo-randomly generated light shows. Therefore one can conclude that the AI approach for composing light shows is not better than the pseudo-random approach used in the control group. Additional research with a larger amount of participants is needed to determine whether people prefer the AI based approach over the pseudo-random approach.

Additionally the judged quality of the AI composed light shows and the human-made light shows has been analyzed on the nine statements separately. For this purpose, t-tests for paired observations have been done based on the difference scores between the human-made light shows and the AI-composed ones and the difference scores between the human-made light shows and the pseudo-randomly generated ones for all statements separately. Results largely match the previous findings when all nine components were combined, and no difference is found between the gain-scores of the AI composed light shows and the pseudo-randomly generated ones that is anywhere close to significant. Somewhat interesting results are

found on two of the nine components, that might suggest that people judge the individual qualities of the AI composed light shows and the pseudo-randomly generated light shows differently.

Firstly, the gain-score of the AI approach is higher than the gain-score of the pseudo-random approach for the statement “The light show was chaotic.” This suggests that the music-similarity algorithm and the algorithm for adapting colors between effects might help in creating a level of order that is closer to the light shows made by a human artist. One must consider however, that the found effect was not statistically significant ($p < 0.2$), therefore no definite conclusions should be drawn from this finding. Additional research with a larger group of participants should be performed to find out whether this effect is truly due to the participants finding the AI composed light shows less chaotic, or whether it was caused by chance.

Secondly, the gain-score of the pseudo-random approach is higher than the gain-score of the AI approach for the statement “There were moments during the light show that I was amazed.” This suggests that using the program for composing light shows decreases the amount of amazement people feel when watching those light shows. Alternatively, it is possible that people interpreted this statement differently than it was intended. Possibly some participants interpreted this statement as “there were moments during the light show that I was *negatively surprised* by the behavior of the light show.” Therefore, the statement should have specified that the participants were “*positively* amazed.” Since the original question was intended as asking in the positive direction, the possibility that participants interpreted this question in the negative sense has not been included in the analysis above. It must be noted that the effect found in the experiment is not statistically significant ($p < 0.1$), and therefore no definite conclusions should be drawn based on this finding. Further research with a larger group of participants should be performed to find out whether this effect is truly due to participants experiencing the pseudo-randomly generated light shows as more amazing than the AI composed ones, or whether it is caused by chance. In further research, this statement should also be rephrased to specify that the participant was *positively* amazed, to reduce the amount of ambiguity.

Lastly, an adaptation on the Feigenbaum test has been performed, as described by Feigenbaum (2003), to see whether people would be able to correctly differentiate

between the behavior of a human expert and an AI on a specific task, namely composing light shows. 45% of participants were able to correctly identify the AI composed light show; 42.86% of the participants were able to correctly identify the pseudo-randomly generated light show. These results seem promising at first. However, the results discussed above show that people prefer the human-made light shows over the light shows composed with the AI or pseudo-random approaches. Additionally many participants have indicated that they found one of the light shows clearly better, but did not know whether this meant that that light show was made by a human or by a computer. What these results show is that people do not know what to expect from a human or a computer when it comes to composing light shows. Therefore, they are unable to judge which light show was made by a computer and which one by a human.

CONCLUSION

This research has proposed a method for composing visual accompaniment to music using Artificial Intelligence techniques. In this research it has been found that light shows made by a human artist are perceived as higher in quality than light shows made by the proposed method. Additionally, it has been found that the program described in this research did not perform significantly better than a pseudo-random approach which selected random visual effects based on the current state of the system. Therefore, it can be concluded that the proposed method will need much improvement before it is capable of composing visual accompaniment to music on the level a human artist would achieve.

ACKNOWLEDGEMENTS

We are happy to acknowledge the help and encouragement of Youtuber *InspirAspir,* for providing us with his own light shows for training and experimenting, as well as helping us formulate questionnaire questions to measure the perceived quality of the light shows.

REFERENCES

Biles, J. A. (1994). GenJam: A genetic algorithm for generating jazz solos. *ICMC Proceedings 1994*, 131-137.

Bolivar, V. J., Cohen, A. J., Fentress, C. (1994). Semantic and Formal Congruency in Music and Motion Pictures: Effects on the Interpretation of Visual Action. *Psychomusicology: Music, Mind & Brain*, *13(1-2)*, 28-59.

De Mantaras, R.L., & Arcos, J.L. (2002). AI and music: From composition to expressive performance. *AI magazine, 23(3)*, 43.

Feigenbaum, E.A. (2003). Some challenges and grand challenges for computational intelligence. *Journal of the ACM, 50(1)*, 32-40.

Hiller, L.A., Isaacson, L.M. (1959). *Experimental music: composition with an electronic computer.* New York: McGraw-Hill Book Company.

Minsky, M (1982). Music, Mind, and Meaning. *Music, Mind, and Brain*, 1-19.

Rader, G.M. (1974). A method for composing simple traditional music by computer. *Communications of the ACM, 17(11)*, 631-638.

Wiggins, G. A. (1998). *Evolutionary methods for musical composition.* Edinburgh: University of Edinburgh, Dept. of Artificial Intelligence.

APPENDIX I

The MIDI code for one event consists of four or more hexadecimal bytes. For example, one MIDI event might look like this:

17 90 33 1
A B C D

The first byte (A) describes the amount of delay in ticks before the MIDI event is executed. For our purposes, we set the amount of time per tick to an arbitrary 1ms/tick. In the example the hexadecimal byte *17* corresponds with a delay of 23 milliseconds.

The second byte (B) describes the type of event. For the launchpad light shows, only “note on” events are used, corresponding to the hexadecimal byte *90*. When a light is turned off, the corresponding MIDI event will code a “note on” event that sets the color of the light to black.

The third byte (C) describes the number of the light that the event codes for. There are 96 lights on the launchpad display numbered 28 (1C) to 123 (7B). In our example, light *33* will be turned on on the launchpad, which corresponds with light number 51.

The fourth byte (D) describes the color that the light will be set to. The launchpad can display 128 predefined colors numbered 0 (0) to 127 (7F). In our example the light is set to color 1 (1), which corresponds with a dark gray.

Whenever a delay longer than 127 milliseconds is required, an additional byte is added to the beginning of the MIDI event. The added bytes are always equal to or larger than 80, whereas the last byte of the delay is always smaller than 80. Whenever the program encounters a delay byte larger than or equal to 80, while playing a light show, 80 is subtracted from this number, and the remainder is multiplied by the maximal amount of time that can be coded by the next byte. To illustrate, the maximal amount of time that can be coded by one delay byte is 127 ticks (7f). If the program needs to wait 128 ticks another byte is added to create (81 00). Similarly, when two bytes are not enough to code the amount of delay required, then a third byte is added to create a delay of 16384 ticks (81 80 00). Up to three additional bytes can be added to the delay, allowing for a massive *FF FF FF 7F*, which codes for

more than an hour of delay.