

Kierunek: **Informatyka Stosowana (IST)**  
Specjalność: **Zastosowania Specjalistycznych Technologii Informatycznych**

**PRACA DYPLOMOWA**  
**MAGISTERSKA**

**Opracowanie algorytmu generacji  
grafu DSP do rozwiązania problemu  
syntezy dźwięku**

**Automated generation of signal  
processing graphs for sound synthesis**

Mateusz Bączek

Opiekun pracy  
**dr inż. Maciej Hojda**

Słowa kluczowe: synteza, dźwięk, graf, optymalizacja



Tekst zawarty w niniejszym szablonie jest udostępniany na licencji Creative Commons: *Uznanie autorstwa – Użycie niekomercyjne – Na tych samych warunkach, 3.0 Polska*, Wrocław 2023.

Oznacza to, że wszystkie przekazane treści można kopiować i wykorzystywać do celów niekomercyjnych, a także tworzyć na ich podstawie utwory zależne pod warunkiem podania autora i nazwy licencjodawcy oraz udzielania na utwory zależne takiej samej licencji. Tekst licencji jest dostępny pod adresem: <http://creativecommons.org/licenses/by-nc-sa/3.0/pl/>.

Licencja nie dotyczy latexowego kodu szablonu. Sam szablon (tj. zbiór przygotowanych komend formatujących dokument) można wykorzystywać bez wzmiankowania o jego autorze. Dlatego podczas redakcji pracy dyplomowej niniejszą stronę można usunąć.

## Streszczenie

Praca prezentuje metodę automatycznej konstrukcji grafu przetwarzania sygnałów dźwiękowych, które wykonują syntezę zadanego przez użytkownika dźwięku. Wytworzony w ramach pracy algorytm może zostać wykorzystany jako narzędzie w pracy inżynierów dźwięku, podczas tworzenia nowych instrumentów elektronicznych lub efektów specjalnych. W przeciwieństwie do technik wykorzystujących sieci neuronowe jako narzędzia syntezy, wynikiem działania algorytmu wytworzonego w ramach pracy jest zrozumiały dla człowieka graf przetwarzania sygnałów, przypominający konwencjonalne konfiguracje syntezy dźwięku wykorzystywane w programach do pracy nad dźwiękiem.

**Słowa kluczowe:** synteza, dźwięk, graf, optymalizacja

## Abstract

TODO!

**Keywords:** synthesis, sound, audio, graph, optimisation

# Spis treści

<b>1. Wstęp . . . . .</b>	<b>9</b>
1.1. Wprowadzenie . . . . .	9
1.2. Cel pracy . . . . .	11
1.2.1. Generowanie grafu przetwarzania sygnałów . . . . .	12
1.2.2. Funkcja celu oceniająca podobieństwo barwy dźwięku . . . . .	13
1.2.3. Problem optymalizacyjny . . . . .	13
1.3. Zakres pracy, plan badań . . . . .	14
1.3.1. Metody generowania grafu przetwarzania sygnałów oraz późniejsza modyfikacja grafu . . . . .	14
1.3.2. Dobór funkcji błędu – różnica między wygenerowanym a docelowym sygnałem dźwiękowym . . . . .	14
1.4. Struktura i zawartość pracy . . . . .	14
<b>2. Graf przetwarzania sygnałów . . . . .</b>	<b>16</b>
2.1. Środowisko . . . . .	16
<b>3. Funkcja celu – porównanie barwy dźwięku . . . . .</b>	<b>17</b>
<b>4. Optymalizacja struktury grafu DSP oraz jego parametrów . . . . .</b>	<b>18</b>
<b>5. Analiza wyników, możliwe drogi dalszego rozwoju . . . . .</b>	<b>19</b>
<b>Literatura . . . . .</b>	<b>20</b>
<b>A. Instrukcja wdrożeniowa . . . . .</b>	<b>22</b>
<b>B. Opis załączonej płyty CD/DVD . . . . .</b>	<b>23</b>

# Spis rysunków

1.1.	Zapis nutowy utworu <i>Opus One</i> , wygenerowany przez komputer <i>Lamus</i> . . . . .	10
1.2.	Przykładowy spektrogram wygenerowany przez algorytm <i>Stable Riffusion</i> dla danych wejściowych <i>funk bassline with a jazzy saxophone solo</i> . . . . .	10
1.3.	Synteza <i>Mother 32</i> firmy <i>Moog</i> , po prawej stronie widoczny jest <i>patch bay</i> z podłączonymi przewodami, które zmieniają konfigurację połączeń między układami generującymi i przetwarzającymi sygnał dźwiękowy. . . . .	11
1.4.	Zbiór parametrów konfiguracyjnych przykładowy syntezy dźwięku w programie <i>Ableton</i> . . . . .	12
1.5.	Diagram blokowy pojedynczego głosu w syntezy <i>Minilogue xd</i> firmy <i>Korg</i> [11]. . . . .	13

# Spis tabel

# Spis listingów

# Skróty

- DAW** (ang. *Digital Audio Workstation*) – oprogramowanie dostępne na komputery osobiste, służące do komponowania utworów muzycznych.
- STFT** (ang. *Short-time Fourier Transform*) – wariant transformaty Fouriera, wykonujący transformację na ruchomym oknie przesuwanym się wzdłuż analizowanego sygnału. **STFT** pozwala na zwiększenie dokładności transformaty dla sygnałów o dużej zmienności w czasie. W kontekście syntezy audio, **STFT** zwiększa dokładność z jaką rejestrowane są transjenty, czyli dynamiczne zmiany charakterystyki barwy dźwięku w czasie.
- CV** (ang. *Control Voltage*) – Sygnał sterujący parametrami syntezy dźwięku, standardowo wykorzystywanych w synteзаторach modułowych (przykładowo w standardzie *EuroRack*). Sygnał **CV** wykorzystuje się do przekazywania sygnałów kontrolnych między modułami.
- VCO** (ang. *Voltage Controlled Oscillator*) – komponent elektroniczny generujący sygnał dźwiękowy. Parametry generowanego sygnału sterowane są za pomocą napięcia kontrolnego (**CV**).
- VCF** (ang. *Voltage Controlled Filter*) – komponent elektroniczny wykonujący filtrację dźwięku w domenie częstotliwości. Parametry filtra sterowane są za pomocą napięcia kontrolnego (**CV**).



# Rozdział 1

## Wstęp

### 1.1. Wprowadzenie

Rozpowszechnione algorytmy sztucznej inteligencji wspomagające pracę inżynierów dźwięku można podzielić na trzy główne kategorie [1] :

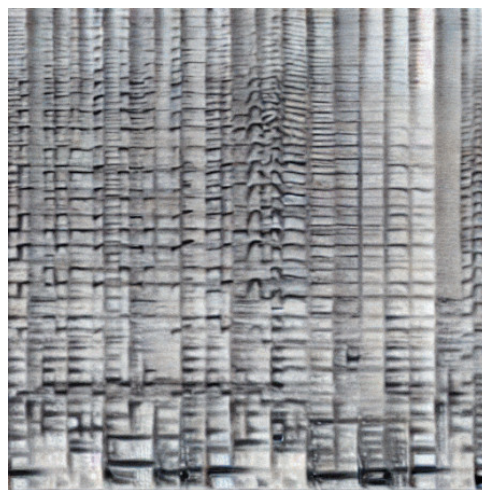
1. algorytmy generujące symboliczny zapis muzyki (nuty lub dane MIDI) (1.1), [18],
2. algorytmy generujące gotowy plik audio (1.2).
3. algorytmy symulujące brzemienie instrumentów muzycznych za pomocą sieci neuronowych [3].

Pierwsza grupa algorytmów znana jest już od lat 80, gdyż zagadnienie generowania zapisu symbolicznego wymaga mniej mocy obliczeniowej niż wytworzenie pełnego pliku audio. Powszechnie wykorzystywana jest w nich teoria muzyki, pozwalająca określić matematyczne relacje występujące w rytmach, melodiach i progresjach akordów. Wiedza dotycząca teorii muzyki pozwala na wyznaczenie możliwej przestrzeni stanów, w której generowana jest kompozycja, natomiast modele matematyczne takie jak łańcuchy Markowa służą za mechanizmy decyzyjne, które „nawigują” w przestrzeni stanów.

Iamus  
Opus #1

Rys. 1.1: Zapis nutowy utworu *Opus One*, wygenerowany przez komputer *Lamus*.

Druga grupa algorytmów, generująca pliki audio, rozwija się na bazie nowych możliwości, które zapewniają algorytmy wywodzące się ze *Stable Diffusion* [14]. Najnowsze modele generujące pliki audio zgodne z opisem tekstowym (przykładowo „smutny jazz” bądź „muzyka taneczna w stylu Depeche Mode”) szkolone są w taki sam sposób jak algorytmy stable diffusion. Jednakże zamiast na obrazach artystów, modele takie jak *Stable Riffusion* [5] uczą się na spektrogramach, które następnie są w stanie wygenerować (1.2). Po wygenerowaniu spektrogramu przez model, jest on konwertowany do pliku audio za pomocą odwrotnej transformaty Fouriera.



Rys. 1.2: Przykładowy spektrogram wygenerowany przez algorytm *Stable Riffusion* dla danych wejściowych funk bassline with a jazzy saxophone solo.

Obie metody opisane w rozdziale 1.1 można porównać pod względem ich przydatności dla użytkownika końcowego, czyli osoby zajmującej się produkcją nagrań muzycznych. Metoda pierwsza, generowanie zapisu symbolicznego, może wydawać się mniej zaawansowana niż generowanie całych plików dźwiękowych. Jednakże, z perspektywy użytkownika, zapis symbo-

liczny jest bardziej praktyczny, ponieważ możliwe jest zaimportowanie go do programu DAW i późniejsza modyfikacja zapisu nutowego. Obecnie dostępne modele generujące pełne nagrania z muzyką nie umożliwiają szczegółowego edytowania parametrów wygenerowanego dźwięku, ponieważ operują bardzo wysokopoziomowo – syntezują muzykę na podstawie opisu słownego.

Podsumowując, wykorzystanie wygenerowanego przez komputer zapisu nutowego jest proste, ze względu na symboliczną naturę zapisu. Wykorzystanie wygenerowanego przez komputer **dźwięku** jest ograniczone ze względu na fakt, że do generowania złożonych sygnałów dźwiękowych wykorzystywane są techniki takie jak głębokie sieci neuronowe, w których utrudniona jest dokładna kontrola nad konkretnymi parametrami funkcjonowania sieci.

Niniejsza praca sugeruje nową metodę podejścia do problemu generowania sygnałów dźwiękowych, którego nie da się zaklasyfikować do żadnej z dwóch wyżej wymienionych (1.1) głównych dziedzin komputerowej kompozycji muzycznej. Wynik pracy algorytmu implementowanego w ramach pracy magisterskiej jest **gotowym elektronicznym instrumentem muzycznym**, który może być wykorzystany w programie do komponowania muzyki. Algorytm nie generuje bezpośrednio sygnału dźwiękowego, lecz tworzy graf przetwarzania sygnałów, który jest zrozumiały dla użytkownika i **pozwala na precyzyjne dostosowanie parametrów syntezy**. Tego typu proces generowania grafów przetwarzania sygnałów dźwiękowych może być porównany z procesem projektowania instrumentu muzycznego.

Modyfikowanie grafu przetwarzania sygnału jest techniką często wykorzystywaną w muzyce elektronicznej, do tworzenia dźwięków o interesującej barwie bądź dynamice. Syntezatory dźwięku dostępne na rynku często wyposażone są w *patch bay*, pozwalający na modyfikowanie grafu przepływu sygnałów wewnątrz syntezatora, bądź połączenie go z zewnętrznym sprzętem muzycznym bądź elektronicznym (1.3).



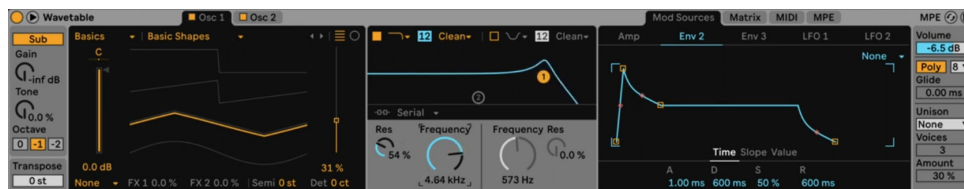
Rys. 1.3: Syntezator *Mother 32* firmy *Moog*, po prawej stronie widoczny jest *patch bay* z podłączonymi przewodami, które zmieniają konfigurację połączeń między układami generującymi i przetwarzającymi sygnał dźwiękowy.

## 1.2. Cel pracy

Celem pracy jest zbadanie, czy algorytmy optymalizacyjne są w stanie wytworzyć graf przetwarzania sygnałów audio, który wykona syntezę próbki dźwięku zadanej przez użytkownika. Problem poruszany w pracy można zakwalifikować do grupy zagadnień związanych z pojęciami

computer-aided design oraz generative artificial intelligence, zastosowanymi w dziedzinie inżynierii dźwięku. Docelowo zaimplementowany algorytm będzie automatyzował pracę inżyniera dźwięku, tworząc i konfiguruje grafy przetwarzania sygnałów dźwiękowych, dostępne w programach typu *digital audio workstation* 1.4. Badania obejmują dwa zagadnienia:

1. metody generowania grafu przetwarzania sygnałów oraz późniejszej modyfikacji grafu – jego struktury oraz parametrów,
2. dobór funkcji celu, na podstawie której algorytm optymalizujący będzie modyfikował graf przetwarzania sygnałów.



Rys. 1.4: Zbiór parametrów konfiguracyjnych przykładowy syntezator dźwięku w programie Ableton

### 1.2.1. Generowanie grafu przetwarzania sygnałów

Pierwsze zagadnienie sprowadza się do przetestowania szeregu algorytmów pozwalających na wygenerowanie grafu przetwarzania sygnałów DSP oraz ich modyfikację. Przykładem modyfikacji grafu może być wprowadzanie w nim losowych zmian lub krzyżowanie dwóch grafów DSP w przypadku wykorzystania algorytmu genetycznego. Graf przetwarzania sygnałów można opisać jako zbiór połączonych węzłów generujących i przetwarzających sygnał dźwiękowy. Każdy węzeł można opisać poprzez:

1. zbiór wejść,
2. zbiór wyjść,
3. operację matematyczną, wykonywaną na sygnale.

Pełny graf przetwarzania można opisać za pomocą zbioru węzłów oraz macierzy połączeń między węzłami:

$N$  - liczba węzłów,

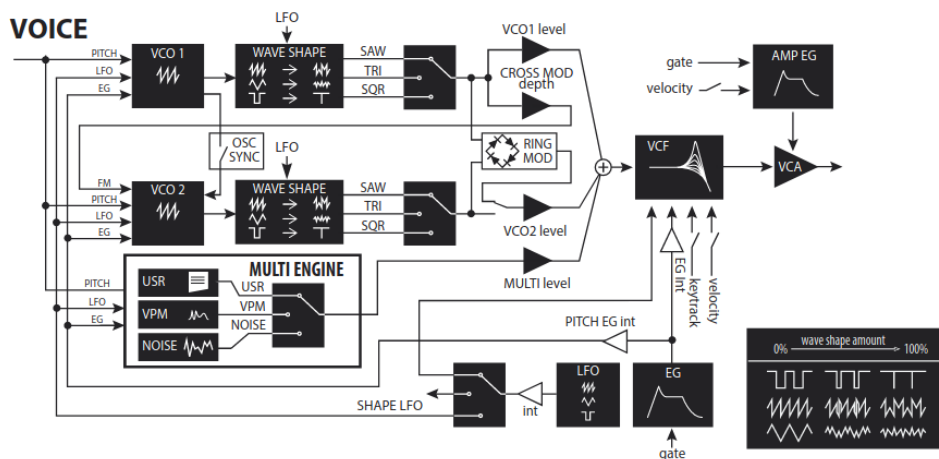
$i_j = [p_1, p_2, \dots, p_n]$  - Zbiór wejść (*inputs*)  $j$ -go węzła,

$o_j = [p_1, p_2, \dots, p_n]$  - Zbiór wyjść (*outputs*)  $j$ -go węzła,

$f_i(x)$  - operacja wykonywana na sygnale przez  $i$ -ty węzeł.

$C = [\{o_{(j,k)}, i_{(l,m)}\}, \dots]$  - zbiór połączeń między węzłami, opisujący, które  $k$ -te wyjście  $j$ -go węzła podłączone jest do którego  $m$ -go wejścia  $l$ -go węzła.

Nie wszystkie wejścia w grafie muszą być podłączone do któregoś z wyjść. Wejście, które nie zostało nigdzie podłączone przyjmuje jako wartość parametr liczbowy optymalizowany później w funkcji celu 1.1. W przypadku schematu 1.5 takimi „wolnymi” wejściami są przykładowo sygnał określający częstotliwość odcięcia filtra sygnału lub parametry określające parametry generatora obwiedni (*EG*).



Rys. 1.5: Diagram blokowy pojedynczego głosu w syntezatorze *Minilogue xd* firmy Korg [11].

Dla powszechnie wykorzystywanego w analogowych syntezatorach subtraktywnych schematu przetwarzania sygnałów (1.5) można wyróżnić przykładowe węzły:

#### Oscylator (VCO):

1. Wejścia:
  - częstotliwość,
  - kształt fali.
2. Wyjścia:
  - wygenerowany sygnał.

#### Filtr (VCF):

1. Wejścia:
  - częstotliwość odcięcia,
  - rezonans.
2. Wyjścia:
  - przefiltrowany sygnał.

### 1.2.2. Funkcja celu oceniająca podobieństwo barwy dźwięku

Drugie zagadnienie obejmuje przetestowanie szeregu algorytmów, które można wykorzystać jako funkcję celu, która będzie optymalizowana poprzez „dostrajanie” grafu przetwarzania sygnałów dźwiękowych.

Funkcja celu, oceniająca, jak sygnał wygenerowany ( $\bar{x}$ ) przez algorytm jest bliski sygnałowi docelowemu ( $x$ ) może zostać przedstawiona w następujący sposób:

$$F(x, \bar{x}) = q \quad (1.1)$$

Gdzie  $q$  oznacza liczbę rzeczywistą, która jest tym mniejsza, im bardziej zbliżone do siebie są barwy dźwięku sygnałów  $x$  i  $\bar{x}$ , przy założeniu że oba sygnały przedstawiają dźwięk o tej samej częstotliwości podstawowej.

### 1.2.3. Problem optymalizacyjny

Następnie, dla danego układu  $N$  węzłów przetwarzania oraz dla macierzy połączeń  $C$ , należy rozwiązać następujący problem optymalizacji, należy rozwiązać problem maksymalizacji funkcji opisaną równaniem 1.1 dla parametrów wszystkich wejść  $i_j$  oraz  $o_j$ , które nie są połączone bezpośrednio pomiędzy węzłami.

## 1.3. Zakres pracy, plan badań

### 1.3.1. Metody generowania grafu przetwarzania sygnałów oraz późniejsza modyfikacja grafu

Głównym problemem przy generowaniu grafu przetwarzania sygnałów są ograniczenia nałożone na strukturę grafu, które należy spełnić, by graf był logicznie interpretowalny jako łańcuch przetwarzania sygnałów. Graf musi być grafem skierowanym, który nie zawiera pętli o dodatnim sprzężeniu zwrotnym (lub nie zawiera ich wcale, co można założyć dla uproszczenia problemu). Struktura grafu powinna być możliwie jak najbardziej przejrzysta dla użytkownika. Automatyczna ewolucja może dążyć w kierunku wykorzystania nadmierowej liczby bloków przetwarzania sygnału, jeśli funkcja celu nie będzie zawierała kary za zbyt złożone grafy. Podobne prace [13] wykorzystują podejście oparte o *mixed-typed cartesian genetic programming*, które będzie punktem startowym dla pracy. Finalnie, badania dążą do wyznaczenia algorytmu o następujący właściwościach:

1. algorytm generuje grafy będące logicznie spójnymi łańcuchami przetwarzania dźwięku (skierowany, bez pętli o dodatnim sprzężeniu zwrotnym w natężeniu sygnału),
2. algorytm maksymalizuje wykorzystanie poszczególnych bloków przetwarzania w grafie, co minimalizuje finalny rozmiar grafu, czyniąc go bardziej czytelnym,
3. generowany graf posiada reprezentację umożliwiającą wykonanie krzyżowania dwóch grafów przetwarzania sygnału. Graf będący wynikiem krzyżowania nadal musi być poprawnym grafem przetwarzania sygnału.

Elementami grafu przetwarzania sygnałów są używane powszechnie w syntezie dźwięku algorytmy:

1. modulacja FM [17] [12],
2. synteza subtraktywna [12] [15],
3. algorytmy *physical modeling* [6] [12],
4. symulacja efektu pogłosu/echa [2] [16].

### 1.3.2. Dobór funkcji błędu – różnica między wygenerowanym a docelowym sygnałem dźwiękowym

Funkcja celu poszukiwana w ramach projektu musi określać, jak dobrze sygnał wygenerowany przez graf przetwarzania sygnałów pokrywa się z sygnałem docelowym. Porównanie sygnałów musi skupiać się na cechach sygnału, które są najbardziej słyszalne dla ludzkiego ucha. Jednocześnie funkcja nie powinna „karać” sygnałów, które są względem siebie przesunięte w fazie. Wśród algorytmów, które zostały wybrane do przetestowania w ramach projektów zawarte są:

1. algorytmy porównywania sygnałów oparte o transformatę Fouriera [7] [19],
2. techniki wykorzystywane do generowania „cyfrowych podpisów” sygnałów dźwiękowych (*sound fingerprinting*) [8],
3. algorytmy wykrywające spadek jakości dźwięku z perspektywy psychoakustycznej [9] [10].

## 1.4. Struktura i zawartość pracy

Praca podzielona jest na następujące części:

## **Środowisko eksperymentalne - graf przetwarzania sygnałów (2)**

Opisuje zaimplementowane w ramach pracy środowisko eksperymentalne, pozwalające na wytwarzanie grafów przetwarzania sygnałów o dowolnej strukturze. Przedstawia zaimplementowane algorytmy syntezy i przetwarzania sygnałów dźwiękowych.

## **Analiza możliwych funkcji celu (3)**

Porównuje funkcje z dziedziny przetwarzania sygnałów, które pozwalają określić jak podobna jest barwa dźwięku dwóch sygnałów dźwiękowych.

## **Optymalizacja struktury grafu DSP oraz jego parametrów (4)**

Opisuje proces badawczy, w którym narzędzia wytworzone w rozdziałach 2 oraz 3 zostały wykorzystane do automatycznego wytworzenia grafu DSP, który naśladuje barwę zadanej próbki dźwięku.

## **Analiza wyników, dyskusja nad skutecznością działania algorytmu oraz możliwe drogi dalszego rozwoju (5)**

Rozdział podsumowuje uzyskane wyniki badań, podejmuje dyskusję nad ogólną skutecznością i przydatnością zaimplementowanego rozwiązania oraz kreśli potencjalne drogi dalszego rozwoju prac badawczych w podobnej tematyce. W czasie, gdy niniejsza praca była tworzona, zostały opublikowane badania dotyczące podobnego problemu [4], rozdział podejmuje dyskusję o różnicach w podejściu do problemu oraz potencjalnych zalet i wad każdego z podejść.

## **Rozdział 2**

# **Graf przetwarzania sygnałów**

### **2.1. Środowisko**



## **Rozdział 3**

# **Funkcja celu – porównanie barwy dźwięku**

## **Rozdział 4**

# **Optymalizacja struktury grafu DSP oraz jego parametrów**

## **Rozdział 5**

# **Analiza wyników, możliwe drogi dalszego rozwoju**

# Literatura

- [1] N. Collins. The analysis of generative music programs. *Organised Sound*, 13(3):237–248, 2008.
- [2] J. Dattorro. Effect design 1: Reverberator and other filters. 1997.
- [3] J. Engel, C. Resnick, A. Roberts, S. Dieleman, D. Eck, K. Simonyan, M. Norouzi. Neural audio synthesis of musical notes with wavenet autoencoders, 2017.
- [4] D. Faronbi, I. Roman, J. P. Bello. Exploring approaches to multi-task automatic synthesizer programming. *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, strony 1–5, 2023.
- [5] S. Forsgren, H. Martiros. Riffusion - Stable diffusion for real-time music generation. 2022.
- [6] N. Hind. *Common Lisp Music (CLM) Tutorials*. wydanie first, 2021. Available for free at <https://ccrma.stanford.edu/software/clm/compmus/clm-tutorials/toc.html>.
- [7] E. Jacobsen, R. Lyons. The sliding dft. *IEEE Signal Processing Magazine*, 20(2):74–80, 2003.
- [8] Y. Ke, D. Hoiem, R. Sukthankar. Computer vision for music identification. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, wolumen 1, strony 597–604 vol. 1, 2005.
- [9] A. F. Khalifeh, A.-K. Al-Tamimi, K. A. Darabkh. Perceptual evaluation of audio quality under lossy networks. *2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, strony 939–943, 2017.
- [10] K. Kilgour, M. Zuluaga, D. Roblek, M. Sharifi. Fréchet audio distance: A metric for evaluating music enhancement algorithms, 2018.
- [11] KORG. Korg minilogue xd user manual, 2019.
- [12] S. Luke. *Computational Music Synthesis*. wydanie first, 2021. Available for free at <http://cs.gmu.edu/~sean/book/synthesis/>.
- [13] M. Macret, P. Pasquier. Automatic design of sound synthesizers as pure data patches using coevolutionary mixed-typed cartesian genetic programming. *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation, GECCO '14*, strona 309–316, New York, NY, USA, 2014. Association for Computing Machinery.
- [14] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, B. Ommer. High-resolution image synthesis with latent diffusion models, 2021.
- [15] J. O. Smith. *Introduction to Digital Filters with Audio Applications*. W3K Publishing, <http://www.w3k.org/books/> <http://www.w3k.org/books/>, 2007.
- [16] J. O. Smith. Physical signal audio processing. 2010.

- 
- [17] J. O. Smith. *Spectral Audio Signal Processing*. <http://ccrma.stanford.edu/ios/sasp/>, accessed <date>. online book, 2011 edition.
- [18] L. Zhang, C. Callison-Burch. Language models are drummers: Drum composition with natural language pre-training, 2023.
- [19] F. Zheng, G. Zhang, Z. Song. Comparison of different implementations of mfcc. *Journal of Computer Science and Technology*, 16(6):582–589, Nov 2001.

## Dodatek A

# Instrukcja wdrożeniowa

Jeśli praca skończyła się wykonaniem jakiegoś oprogramowania, to w dodatku powinna pojawić się instrukcja wdrożeniowa (o tym jak skompilować/zainstalować to oprogramowanie). Przydałoby się również krótkie „*how to*” (jak uruchomić system i coś w nim zrobić – zademonstrowane na jakimś najprostszym przypadku użycia). Można z tego zrobić osobny dodatek.

cargo run i jazda

## Dodatek B

# Opis załączonej płyty CD/DVD

Tutaj jest miejsce na zamieszczenie opisu zawartości załączonej płyty. Opis ten jest redagowany przed załadowaniem pracy do systemu APD USOS, a więc w chwili, gdy nieznana jest jeszcze nazwa, jaką system ten wygeneruje dla załadowanego pliku. Dlatego też redagując treść tego dodatku dobrze jest stosować ogólniki typu: „Na płycie zamieszczono dokument pdf z niniejszej tekstem pracy” – bez wskazywania nazwy tego pliku.

Dawniej obowiązywała reguła, by nazywać dokumenty według wzorca W04\_[nr albumu]\_[rok kalendarzowy]\_[rodzaj pracy], gdzie rok kalendarzowy odnosił się do roku realizacji kursu „Praca dyplomowa”, a nie roku obrony. Przykładowo wzorzec nazwy dla pracy dyplomowej inżynierskiej w konkretnym przypadku wyglądał tak: W04\_123456\_2015\_praca inżynierska.pdf, Takie nazwy utrwalane były w systemie składania prac dyplomowych. Obecnie działa to już inaczej.