Generator Musik Menggunakan *Convex Hull*

Nathan Tenka – 13521172
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13521172@std.stei.itb.ac.id

***Abstrak*—*Convex hull* adalah kumpulan titik *convex* terkecil yang mengandung seluruh himpunan titik tertentu. *Convex hull* umumnya digunakan untuk statistik, *clustering*, pengolahan citra, dan bidang-bidang lain. *Convex hull* juga bisa digunakan dalam bidang *algorithmic composition* untuk membuat musik dari data. Musik dihasilkan menggunakan dua metode pemetaan, yaitu pemetaan rentang dan modulo. Melodi digenerasi secara acak sehingga *convex hull* yang sama bisa menghasilkan musik yang berbeda. Musik yang dihasilkan dipengaruhi oleh koordinat titik-titik *convex hull dan* cenderung memiliki pengulangan not dan *pitch*. Tidak ditemukan parameter yang bisa memprediksi kunci, *chord progression*, dan *mood* dari musik yang dihasilkan.**

Kata Kunci*—*convex hull, algorithmic composition, musik, graham scan

I. PENDAHULUAN

Salah satu kegiatan yang dapat dibantu menggunakan komputer adalah membuat musik. Sejak munculnya teori musik formal, hubungan antara musik dengan fenomena alam dan konsep-konsep matematis telah dipelajari secara mendalam. Dengan kedatangan komputer, muncul metode-metode pembuatan musik yang memanfaatkan algoritme yang dijalankan oleh komputer. Sekarang, ada banyak komposer yang menggunakan musik dari komputer sebagai inspirasi atau bahkan langsung dirilis.

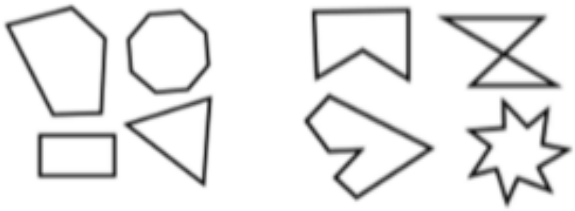
Ada berbagai cara yang bisa digunakan komputer untuk membuat musik secara otomatis. Ada yang menggunakan data acak yang kemudian diubah menjadi not, ada juga yang menggunakan model matematis seperti fraktal. Selain itu, ada juga model kecerdasan buatan yang dapat membuat musik sendiri.

Beberapa musik yang dihasilkan dari komputer ini berhasil menjadi musik yang ramai didengarkan orang. Contohnya adalah Holly Herndon & Jlin (feat. Spawn) – *Godmother* yang dibuat menggunakan kecerdasan buatan bernama Spawn dan Brian Eno : *Reflection* [4]. Dalam makalah ini, penulis akan menguji pembuatan musik menggunakan *convex hull*.

II. LANDASAN TEORI

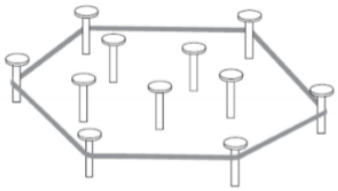
A. *Convex Hull*

Suatu kumpulan titik pada bidang planar disebut *convex* jika untuk sembarang dua titik pada bidang (misal p dan q), seluruh segmen garis yang berakhir di p dan q berada pada himpunan tersebut [1]. Secara visual, poligon *convex* tidak memiliki bagian yang cekung ke dalam. Contoh poligon yang *convex* dan tidak *convex* ada pada kedua gambar berikut :



Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Divide-and-Conquer-(2022)-Bagian4.pdf
Gambar 2.1 dan 2.2 Contoh poligon *convex* (kiri) dan tidak *convex* (kanan)

Convex hull dari suatu kumpulan titik adalah himpunan *convex* terkecil yang mengandung seluruh kumpulan titik tersebut [1].



Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Divide-and-Conquer-(2022)-Bagian4.pdf
Gambar 2.3 Contoh *convex hull*

B. *Algoritme Greedy*

Algoritme *greedy* adalah algoritme yang memecahkan suatu persoalan dengan memilih pilihan yang terbaik pada setiap langkah. Algoritme *greedy* mengharapkan hasil yang didapat adalah yang terbaik secara keseluruhan (memilih optimum lokal untuk mendapat optimum global). Algoritme *greedy* terdiri dari beberapa elemen sebagai berikut [2] :

- Himpunan kandidat, C : berisi kandidat yang akan dipilih pada setiap langkah
- Himpunan solusi, S : berisi kandidat yang sudah dipilih
- Fungsi solusi : menentukan apakah himpunan kandidat yang dipilih sudah memberikan solusi
- Fungsi seleksi : memilih kandidat berdasarkan strategi *greedy* tertentu. Strategi bersifat heuristik
- Fungsi kelayakan : memeriksa apakah kandidat yang dipilih bisa dimasukkan dalam himpunan solusi

- Fungsi obyektif : meminimumkan atau memaksimumkan

C. *Algoritme Graham Scan*

Algoritme Graham Scan adalah salah satu algoritme pembuatan *convex hull* yang lebih efisien dibanding algoritme-algoritme yang lain. Langkah-langkah dalam algoritme Graham Scan adalah sebagai berikut [8] :

1. Cari titik terbawah (memiliki posisi y terkecil) dari himpunan titik yang diberikan. Apabila ada lebih dari satu titik dengan posisi y yang sama, ambil titik dengan posisi x yang lebih kecil. Misalkan titik ini adalah titik P0. Masukkan P0 ke dalam himpunan titik *convex hull* di posisi pertama.
2. Urutkan titik-titik lain berdasarkan sudut polar berlawanan arah jarum jam dengan titik P0 dari kecil ke besar. Jika ada lebih dari satu titik dengan sudut polar yang sama, urutkan berdasarkan jarak dengan titik P0.
3. Setelah mengurutkan, periksa sudut pada setiap titik. Apabila ada lebih dari satu titik dengan sudut yang sama, hapus semua titik tersebut kecuali titik yang terjauh dari P0. Misalkan ukuran senarai yang baru adalah m.
4. Jika m < 3, tuliskan "*Convex hull* tidak bisa dibuat".
5. Buat *stack* kosong (misalkan S) dan *push* tiga titik pertama ke dalam S.
6. Proses m-3 titik yang lain. Untuk tiap titik :
   6.1. Hapus titik dari S selama orientasi 3 titik berikut searah jarum jam :
      a. Titik kedua teratas di S.
      b. Titik teratas di S.
      c. Titik ke-i (titik yang sedang dikunjungi)
   6.2. *Push* titik ke-i ke dalam S.
7. S akan berisi kumpulan titik *convex hull*.

Secara keseluruhan, algoritme ini terdiri dari 2 fase :

1. Fase mengurutkan titik-titik : semua titik diurutkan terlebih dahulu berdasarkan sudutnya dengan titik terbawah. Titik-titik yang telah diurutkan ini akan membentuk jalur tertutup yang sederhana. Tetapi, karena menghitung sudut antar titik memerlukan komputasi yang "mahal", perbandingan sudut bisa digantikan dengan membandingkan orientasi antar titik.
2. Menerima atau menolak titik : setelah didapat jalur tertutup, telusuri jalur tersebut dan hapus semua titik yang menyebabkan poligon *concave*. Titik *concave* bisa dicari dengan membandingkan orientasi antara 3 titik "terbaru" dalam jalur tersebut. Jika orientasi yang dibentuk ketiga titik tidak berlawanan arah jarum jam, berarti titik yang sedang dikunjungi merupakan titik *concave* dan bisa dihapus dari jalur.

Kompleksitas waktu algoritme ini bisa dihitung sebagai berikut :

a. Pencarian titik terbawah memerlukan waktu O(n).
b. Perhitungan orientasi tiap titik terhadap titik terbawah memerlukan waktu O(n).
c. Pengurutan titik berdasarkan sudut polar dengan titik terbawah memerlukan waktu $O(n^2)$ atau O(n log n), tergantung algoritme pengurutan yang digunakan.
d. Pemeriksaan orientasi tiap titik memerlukan waktu O(n).

Pengurutan titik akan dilakukan menggunakan algoritme *quicksort*, sehingga kompleksitas waktu algoritme Graham Scan adalah O(n) + O(n) + O(n log n) + O(n) = O(n log n).

D. *Algorithmic Composition*

Algorithmic composition adalah istilah untuk pembuatan musik oleh komputer menggunakan algoritme tertentu. Ada beberapa metode yang bisa digunakan dalam *algorithmic composition* [6], yaitu

- *Mathematical models*
- *Knowledge based systems*
- *Grammars*
- *Evolutionary methods*
- *Systems which learn*
- *Hybrid systems*

Selain keenam metode di atas, ada juga metode *translational model* yang menerjemahkan informasi menjadi musik. Pendekatan yang diterapkan dalam makalah ini termasuk dalam *translational model* karena dilakukan "translasi" dari *convex hull* menjadi musik.

E. *Teori Musik*

Teori musik adalah ilmu yang mempelajari konsep dasar di balik musik dan menyediakan cara formal untuk menginterpretasi musik. Teori musik merupakan bidang yang kompleks. Oleh karena itu, dalam makalah ini hanya akan dijelaskan teori musik dasar yang digunakan. Ada 3 komponen dasar dalam sebuah komposisi musik [5], yaitu

1. Harmoni

   Harmoni adalah peristiwa saat beberapa not atau suara dimainkan secara bersamaan sehingga menghasilkan bunyi baru. Contoh harmoni adalah *chord* dan *chord progression*. *Chord* adalah kumpulan tiga atau lebih not yang dimainkan secara bersamaan, sedangkan *chord progression* adalah serangkaian *chord* yang dimainkan secara berurutan. Ada dua jenis utama harmoni, yaitu *dissonant* dan *consonant*.

- *Dissonant harmony* adalah kumpulan suara yang tidak enak didengar secara bersamaan. Contoh interval *dissonant* adalah *seconds, sevenths,* dan *ninths*
- *Consonant harmony* adalah kumpulan suara yang enak didengar secara bersamaan. Contoh interval *consonant* adalah *unison, thirds, fifths,* dan oktaf

2. Melodi

   Melodi adalah kumpulan not atau suara yang disusun sehingga enak didengar. Biasanya melodi merupakan bagian musik yang paling mudah diingat. Dua elemen utama melodi adalah :

   - *Pitch* adalah getaran suara yang dihasilkan oleh instrumen atau sumber lain. *Pitch* menentukan tinggi/rendahnya sebuah not
   - Irama menyatakan berapa lama sebuah nada akan dimainkan. Contoh irama adalah *whole note, half note,* dan *quarter note*

   Selain itu, melodi juga memiliki dua jenis gerakan :

   - *Conjunct motion* terjadi saat perpindahan antar not hanya sebesar satu atau setengah *step* (tidak jauh)
   - *Disjunct motion* terjadi saat perpindahan antar not cukup besar

3. Irama

   Selain pengertian yang sudah disebutkan sebelumnya, irama juga memiliki pengertian lain, yaitu

   - Gerakan berulang not dan *rests* dalam waktu
   - Pola kuat lemah not yang berulang

   Irama terdiri dari :

   - *Beat* : bunyi teratur yang menjadi dasar sebuah pola musik
   - *Meter* : Pola detak kuat dan lemah yang spesifik
   - *Time signature* : jumlah *beat* per suatu pengukuran
   - Tempo (BPM) : menyatakan kecepatan memainkan musik
   - *Strong and weak beats* : *strong beat* adalah bagian utama lagu, sedangkan *weak beat* adalah irama pengiring di antara *strong beat*
   - *Syncopation* : irama yang menonjolkan irama pengiring
   - *Accents* : penekanan pada tiap not

Selain tiga komponen di atas, juga terdapat skala musik dan kunci. Skala musik adalah kumpulan not dalam satu oktaf yang diurutkan berdasarkan *pitch*. Skala musik membentuk melodi dan harmoni. Kunci menandakan skala yang digunakan oleh sebuah komposisi musik. Contohnya, kunci C menandakan lagu menggunakan not dalam skala C.

Musical modes adalah skala yang diturunkan dari sebuah skala utama. Ada tujuh *musical modes* [5], yaitu

- I – Ionian (skala mayor)
- ii – Dorian (skala mayor dimulai dari *2nd degree*)
- iii – Phrygian (skala mayor dimulai dari *3rd degree*)
- IV – Lydian (skala mayor dimulai dari *4th degree*)
- V – Mixolydian (skala mayor dimulai dari *5th degree*)
- vi – Aeolian (skala minor alami atau skala mayor dimulai dari *6th degree*)
- vii – Locrian (skala mayor dimulai dari *7th degree*)

III. Implementasi

A. Pembuatan Convex Hull

Convex hull dibuat menggunakan algoritma Graham Scan seperti yang dijelaskan pada landasan teori. Program akan menerima sebuah file csv. Kemudian, pengguna memilih 2 kolom data numerik untuk digunakan sebagai sumbu penentu not (sumbu x) dan sumbu penentu *pitch* (sumbu y). Program akan membuat *convex hull* dari kedua kolom data tersebut. *Pseudocode* pembuatan *convex hull* memakai algoritme Graham Scan adalah sebagai berikut :

```
function makeConvexHull(coordinates : himpunan_titik)
{ Menghasilkan convex hull dari himpunan titik yang diberikan }
Deklarasi
    lowest_point : titik terendah dari coordinates
    S : stack_titik
Algoritma
    lowest_point ← searchLowestPoint(coordinates)
    sortPolarAngle(coordinates, lowest_point)
    deleteSameAngle(coordinates)
    if size(coordinates) < 2 then
        raise error
    push(S, coordinates[0])
    push(S, coordinates[1])
    filterClockwise(S, coordinates)
    return S
```

Algoritme Graham Scan bisa dikategorikan sebagai algoritme *greedy* karena algoritme ini memilih titik yang membentuk sudut polar terkecil dengan titik terendah. Lalu, algoritme memeriksa orientasi yang dibentuk masing-masing titik secara berurut berdasarkan sudut. Dalam hal ini algoritme Graham Scan memilih optimum lokal pada setiap langkah. Elemen algoritme *greedy* dalam algoritme Graham Scan adalah

- Himpunan kandidat : himpunan sembarang titik
- Himpunan solusi : himpunan titik yang membentuk *convex hull*
- Fungsi solusi : semua titik berurutan yang terpilih tidak membentuk orientasi searah jarum jam
- Fungsi seleksi : pilih titik yang memiliki sudut polar terkecil dengan titik terbawah
- Fungsi kelayakan : orientasi titik dengan dua titik sebelumnya harus berlawanan arah jarum jam
- Fungsi obyektif : jumlah titik yang digunakan minimum

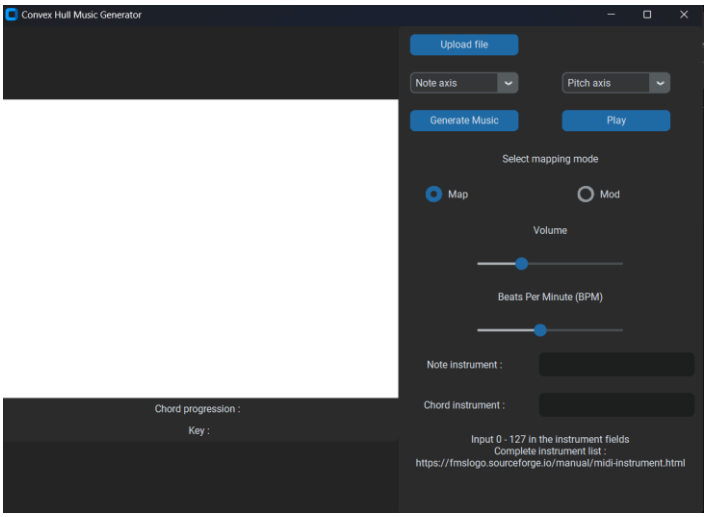
B. Translasi Convex Hull Menjadi Musik

Titik-titik dalam *convex hull* dikonversi menjadi not dengan memanfaatkan *library* mingus. Mingus adalah *library* Python yang menyediakan fungsi dan struktur data yang berkaitan dengan teori musik, seperti not, *bar*, *chord*, dll.

Setelah *convex hull* dibuat, koordinat sumbu x tiap titik dipetakan ke rentang 0-11 dan diubah menjadi not menggunakan *library* Mingus. Koordinat sumbu y dipetakan menjadi *pitch* melodi. Translasi bisa dilakukan dengan pemetaan rentang ataupun modulo. Kumpulan not yang dihasilkan dipakai untuk menentukan kunci musik. Lalu, dari setiap not akan dibuat sebuah *chord* yang dimulai dari not tersebut dengan kunci yang sesuai. Not melodi ditentukan secara acak dari tiap *chord*. Durasi tiap not melodi dan *chord* juga ditentukan secara acak. Not melodi bisa berupa *half note*, *quarter note*, atau *eight note*, sedangkan *chord* bisa berupa *whole note*, *half note*, atau *quarter note*. Musik akan disimpan sebagai fiile Result.mid dalam *folder* program, dan bisa dimainkan dengan menekan tombol "Play". Volume, tempo (dalam BPM), dan instrumen yang digunakan ditentukan oleh pengguna. *Pseudocode* proses translasi *convex hull* adalah sebagai berikut :

```
procedure hullToMusic(input hull : himpunan_titik)
{ Mengubah convex hull menjadi kumpulan not
  Masukkan : kumpulan titik yang membentuk convex hull
  Keluaran : file yang berisi kumpulan not hasil translasi }
Deklarasi
    list_note : kumpulan not hasil translasi
    list_chord : kumpulan chord hasil translasi
    list_melody : kumpulan not melodi
    key : kunci musik yang dihasilkan
Algoritma
    for c in hull :
        insert (translateToNote(c), list_note)
    key ← getKeyFromNotes(list_note)
    for note in list_note :
        current_chord ← getChord(note, key)
        insert(random(current_chord), list_melody)
        insert(current_chord, list_chord)
    write_to_file (list_melody, list_chord)
```

Tampilan antarmuka program seperti berikut :



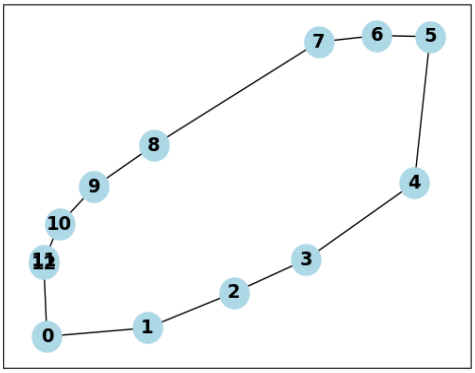
Gambar 3.1 Tampilan antarmuka program

IV. PENGUJIAN DAN ANALISIS

Seluruh pengujian dilakukan memakai data dalam file weather.csv (sumber : https://www.kaggle.com/datasets/zaraavagyan/weathercsv) yang tersedia di dalam folder data *repository* github yang dilampirkan.

A. Pengujian

1. Kolom MinTemp sebagai *note axis* dan MaxTemp sebagai *pitch axis*



Gambar 4.1 *Convex hull* uji kasus 1

Hasil dengan pemetaan rentang :

Chord progression : I – ii -IV -V – vii – vi – V - #Idim – I – I – I

Key : C

Not : ['C', 'D', 'F', 'G', 'A#', 'B', 'A', 'G', 'D#', 'C#', 'C', 'C', 'C']

Pitch : [3, 3, 3, 4, 5, 7, 8, 7, 6, 5, 4, 4, 4]

Tautan hasil :

https://drive.google.com/file/d/1XDRrJuxq2LVQj19pdPdePr9bATXFpED9/view?usp=sharing

Hasil dengan modulo :

Chord progression : #Vdim – ii - #Vdim - #Idim – V - #Vdim – IV - #Idim – vii – vi - #Vdim - #Vdim

Key : C

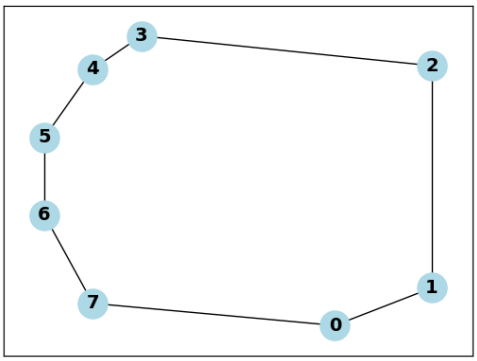
Not : ['G#', 'D', 'G#', 'C#', 'G', 'G#', 'F', 'C#', 'D#', 'B', 'A', 'G#', 'G#']

Pitch : [4, 5, 8, 5, 7, 8, 8, 8, 4, 6, 3, 5, 5]

Tautan hasil :

https://drive.google.com/file/d/167uiNOkZEOyv6kJPjc05vNPKYn4lpE8B/view?usp=sharing

2. Kolom Cloud3pm sebagai *note axis* dan MinTemp sebagai *pitch axis*



Gambar 4.2 *Convex hull* uji kasus 2

Hasil dengan pemetaan rentang :

Chord progression : vii – ii – ii – IV – iii – biiiaug – biiiaug – iii

Key : A

Not : ['G#', 'B', 'B', 'D', 'C#', 'C', 'C', 'C#']

Pitch : [3, 3, 7, 8, 7, 6, 4, 3]

Tautan hasil :

https://drive.google.com/file/d/1c0T1A1iXSgXulpnmNkMADVGLMy2y5ogV/view?usp=sharing

Hasil dengan modulo :

Chord progression : vi – vii – vii – IV – iii – biiiaug – biiiaug – iii

Key : A

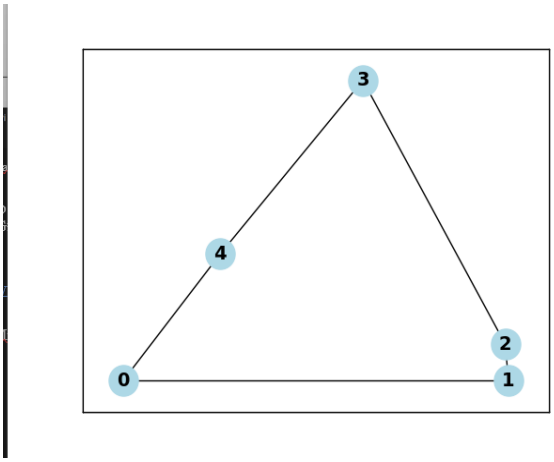
Not : ['F#', 'G#', 'G#', 'D', 'C#', 'C', 'C', 'C#']

Pitch : [3, 7, 3, 5, 8, 8, 7, 5]

Tautan hasil :

https://drive.google.com/file/d/185Eu7vQJ2W1JCmcUe5JmvrjDIpzWBdFX/view?usp=sharing

3. Kolom Temp9am sebagai *note axis* dan RISK_MM sebagai *pitch axis*



Gambar 4.3 *Convex hull* uji kasus 3

Hasil dengan pemetaan rentang :

Chord progression : IV – iii – vii – V

Key : G

Not : ['C', 'B', 'A#', 'F#', 'D']

Pitch : [3, 3, 3, 8, 5]

Tautan hasil :

https://drive.google.com/file/d/1SEBG1rb5rNA7Du6EWnMPRSt3HLyEmllx/view?usp=sharing

Hasil dengan modulo :

Chord progression : IV – IV –IV – #Vdim – vii

Key : G

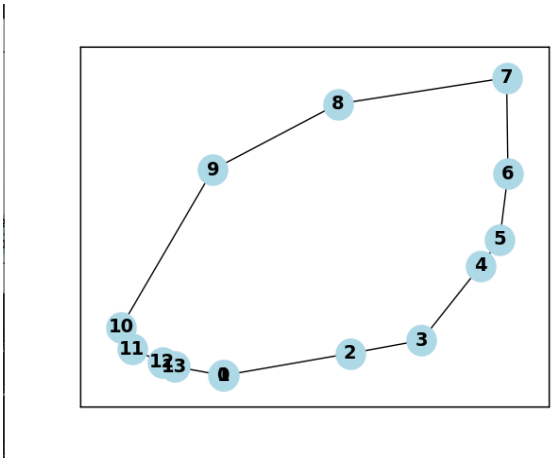
Not : ['C', 'C', 'C', 'D#', 'F#']

Pitch : [3, 3, 7, 6, 7]

Tautan hasil :

https://drive.google.com/file/d/1e1GzWB1UziHVFOwY2IVqtdT_etuyTb3o/view?usp=sharing

4. Kolom MaxTemp sebagai *note axis* dan Evaporation sebagai *pitch axis*



Gambar 4.4 *Convex hull* uji kasus 4

Hasil dengan pemetaan rentang :

Chord progression : IV – IV – vi – vii - #Idim - #Idim – ii - #Idim – vi – IV – biiiaug – biiiaug – iii – iii

Key : A

Not : ['D', 'D', 'F#', 'G#', 'A#', 'A#', 'B', 'A#', 'F#', 'D', 'C', 'C', 'C#', 'C#']

Pitch : [3, 3, 3, 3, 4, 5, 6, 8, 7, 6, 3, 3, 3, 3]

Tautan hasil :

https://drive.google.com/file/d/1DX2mG0OaAAm1IbTRZVIB7xeuByPOG2qQ/view?usp=sharing

Hasil dengan modulo :

Chord progression : I – IV – vi – vii – vii – vii – vii – ii – V - #Vdim – vii

Key : C

Not : ['D#', 'D#', 'C', 'F', 'A', 'B', 'B', 'B', 'B', 'D', 'G', 'G#', 'A#', 'B']

Pitch : [3, 3, 4, 4, 8, 3, 6, 4, 3, 6, 5, 4, 3, 3]

Tautan hasil :

https://drive.google.com/file/d/1TQdI44TtDNrJ6k_082btT0J0tL-e4UpA/view?usp=sharing

B. Analisis

Secara keseluruhan, musik yang dihasilkan memiliki pola yang ditentukan oleh bentuk *convex hull* terutama untuk metode pemetaan rentang. Pada musik yang dihasilkan dengan pemetaan rentang, perkembangan not dan *pitch* mengikuti koordinat titik pada *convex hull* dan perpindahan antar not dan *pitch* relatif teratur berdasarkan posisi titik-titik *convex hull*. Posisi titik yang berjauhan akan menyebabkan perpindahan not dan *pitch* yang besar, begitu pula sebaliknya. Musik yang dihasilkan menggunakan modulo cenderung memiliki perpindahan not dan *pitch* yang lebih tidak teratur.

Tidak ada parameter yang bisa memprediksi dengan pasti kunci, *chord progression*, maupun *mood* yang dihasilkan. Kedua metode cenderung memiliki pengulangan not maupun *pitch*, namun pengulangan pada metode modulo lebih tidak teratur. Untuk musik hasil pemetaan, pengulangan hanya terjadi apabila ada beberapa titik yang lokasinya berdekatan. Selain itu, kedua metode bisa menghasilkan musik dalam kunci yang berbeda, seperti pada uji kasus keempat.

V. KESIMPULAN DAN SARAN

A. Kesimpulan

Convex hull adalah poligon *convex* terkecil yang mengandung seluruh himpunan titik yang lain dalam poligon tersebut. Selain untuk kegunaan seperti *clustering*, *convex hull* juga bisa digunakan untuk komposisi algoritmik (*algorithmic composition*). Setiap titik dalam *convex hull* bisa di-"translasi" menjadi not sehingga bisa membuat beragam jenis musik.

B. *Saran*

Untuk pengembangan lebih lanjut, bisa ditambahkan parameter lain untuk membuat musik lebih bervariasi. Misalnya, tempo bisa ditentukan oleh gradien antar garis atau jumlah titik. Selain itu, bisa juga digunakan kecerdasan buatan untuk membuat musik lebih enak didengar.

VI. Ucapan Terima Kasih

Puji syukur penulis ucapkan kepada Tuhan Yang Maha Esa atas rahmat-Nya sehingga makalah ini bisa diselesaikan tepat waktu. Penulis juga berterima kasih kepada Ibu Dr. Nur Ulfa Maulidevi, S.T, M.Sc. sebagai dosen pengampu mata kuliah IF2211 Strategi Algoritma kelas 02 atas bimbingannya dalam mata kuliah ini. Terakhir, penulis juga berterima kasih kepada kedua orang tua yang selalu memberikan dukungan kepada penulis.

VII. Lampiran

Tautan *repository* github :

https://github.com/Nat10k/ConvexHullMusicGenerator

Tautan Video Youtube

https://youtu.be/KrLNsokIcJ8

Referensi

[1] Munir, Rinaldi. 2021. Algoritma *Divide and Conquer* (Bagian 4). Diakses pada 21 Mei 2023 dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Divide-and-Conquer-(2022)-Bagian4.pdf.

[2] Munir, Rinaldi. 2021. Algoritma *Greedy* (Bagian 1). Diakses pada 21 Mei 2023 dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf.

[3] Spaans, Bart. 2015. mingus. Diakses pada 16 Mei 2023 dari https://bspaans.github.io/python-mingus/.

[4] Google Arts & Culture. 12 songs created by AI. Diakses pada 21 Mei 2023 dari https://artsandculture.google.com/story/12-songs-created-by-ai-barbican-centre/VwVhbAD7QslgLA?hl=en.

[5] PQ, Rory. 2020. BASIC MUSIC THEORY FOR BEGINNERS – THE COMPLETE GUIDE. Diakses pada 21 Mei 2023 dari https://iconcollective.edu/basic-music-theory/.

[6] Papadopoulos, George & Geraint Wiggins. (1999). *AI Methods for Algorithmic Composition: A Survey, a Critical View and Future Prospects*. School of Artificial Intelligence, Division of Informatics, University of Edinburgh. Diakses 21 Mei 2023 dari http://www.doc.gold.ac.uk/~mas02gw/papers/AISB99b.pdf.

[7] Aarti_Rathi. 2023. Convex Hull using Graham Scan. Diakses pada 21 Mei 2023 dari https://www.geeksforgeeks.org/convex-hull-using-graham-scan/.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 22 Mei 2023

Nathan Tenka 13521172