生成 AI を用いた Processing による映像制作補助に関する研究

大森裕太朗[1,a)]　赤池英夫[1,b)]

概要：近年，SNS や動画配信サービスの普及により，視覚的にインパクトのある映像作品の需要が高まる一方，その制作プロセスは初心者にとって大きな障壁となっている．本研究では，音楽を基に背景や映像表現を生成し，Processing のコードとして出力するツールの開発と評価を行う．このツールは，Python ライブラリ Librosa を用いた音楽信号の解析結果を生成 AI である ChatGPT に渡し，適切な映像表現を自動生成する．さらに，ユーザーが修正を加えやすい形で Processing のコードとして出力することで，映像制作の柔軟性を向上させる．評価実験では，初心者が提案システムを使用した場合の制作時間と映像の品質を，従来の動画編集ソフトと比較し，その有効性を検証する．本ツールは，生成 AI を活用した映像制作プロセスの効率化と初心者への敷居の低減を目指している．

1. 背景

近年，SNS や動画配信サービスの普及によって，視覚的にインパクトのある映像作品は情報伝達や広告手段として必要不可欠な存在となった．しかし，映像作品の作成は覚えることの多さやツールの操作の複雑さといった要因によりかなり始めるためのハードルが高いものとなっており，初心者にとって気軽に始めにくく，挫折しやすいものとなってしまっている．近年では，生成 AI や自動化ツールの活用でこのプロセスを簡素化する動きがあり，その中でも特に注目されているのが，生成 AI によるプロンプトベースの映像生成である．プロンプトの記述によって視覚的表現を自動で生成する技術は，クリエイターの創造力をサポートする新たな手段として期待されている．図 1 は OpenAI が 2024 年 12 月 9 日にリリースした映像生成 AI，Sora の公式ページ [1] のサンプルである．これを用いることでプロンプトによって高品質な映像を生成できる．

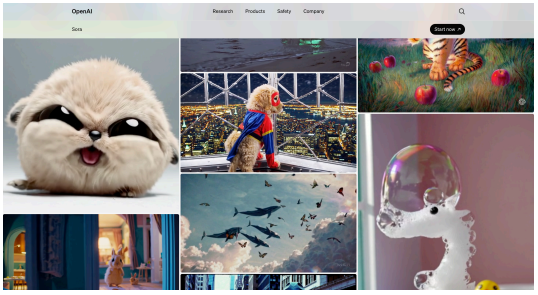

図 1　Sora の公式ページより

2. 関連研究

Vivian liu ら [2] は，大規模な言語モデルとテキストから音楽を視覚化したビデオを生成する生成 AI システムである Generative Disco を開発した．このシステムでは出力として音楽を視覚化した映像が得られるが，後から映像の一部を変更するといったことはできない．本研究の目的は，あくまで映像制作の補助であるため，後から映像の変更を行うことができるよう Processing のコードとして出力を得るようにした．

中渡瀬 [3] の研究では，学術文書の情報構造認識を ChatGPT に行わせた際に，様々な技法でプロンプトの改良を行うことによって出力結果にどのような影響が出るかが検証されていた．本研究では，中渡瀬の研究で用いられた技法のうち，タスクを分割した指示を参考にして実際にプロンプトの作成を行った．

Weiheng ら [4] の研究では，リアルタイムで更新可能な非推奨 API データセットを用いてそれらをコードから除外することで非推奨 API を含まないコード生成を支援する APILOT という手法が提案された．本研究では非推奨 API データセットの作成手法を参考にしてエラーデータセットの作成とそれを用いたエラー再発予防を行った．

[1] 電気通信大学
[a)] o2010126@gl.cc.uec.ac.jp
[b)] xakaike@cs.uec.ac.jp

3. 研究概要

本研究では，入力として音楽を生成 AI に与えることにより，音楽に合った背景，映像表現などを Processing のコードとして出力するツールの開発と評価を行う．対象とする音楽として，まずは歌詞がなく曲の展開がシンプルなものを選択する．概要は図 2 の通り．

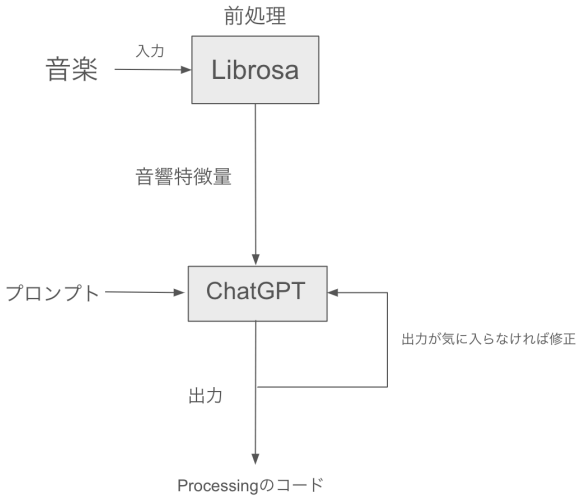

図 2 ツールの概要図

4. 提案手法

まず Python の音楽分析ライブラリ Librosa を用いてデータの前処理を行うことにより，音楽の BPM やオーディオスペクトラム，テンポグラムといった映像生成に必要なデータを抜き出す．そのデータを ChatGPT に渡すことで音楽に適した映像表現の候補を得る．それらの候補のうち使いたい表現について，表現の詳細をプロンプトとして渡すことで，Processing のコードを出力として得る．映像に修正したい箇所があれば追加の指示を出してコードの修正を行う．また，生成されたコードにエラーがあった際に，発生したエラーを抜き出してエラーリストに追加し，次の生成の際にリストを ChatGPT に渡すことで同じエラーが発生するのを防ぐ．ここで，出力を Processing のコードにするのは，AfterEffects や AviUtl などの専用のソフトによって映像制作を行う必要のあるファイルは，外からファイルを書き換えることが困難なためである．生成 AI には自然言語処理とコード生成の両方が可能である ChatGPT を用いる．以上をまとめて Web アプリの形として提案システムを実現する．

4.1 使用している ChatGPT について

ChatGPT は，OpenAI が開発した大規模な自然言語処理モデルで，特に会話形式でのタスクに特化している．基盤には「GPT（Generative Pre-trained Transformer）」という技術が使用されている．この技術は，大量のテキストデータを基に事前学習を行い，文章生成，質問応答，翻訳，文章要約などの幅広いタスクに対応している．ChatGPT にはいくつかのバージョンが用意されており，それぞれ文章生成能力や対応できるタスクなどに違いがある．

GPT-3.5 高性能な基本モデルで，文章生成や簡易タスクに優れるが，長い文脈や高度な推論では限界あり。コスト効率が良い．

GPT-4.0 文脈理解や精度が大幅に向上し，複雑なタスクや長文対応に最適．論理的推論や専門性も強化され，高精度な回答が可能．

GPT-4.0-Turbo GPT-4 と同等の性能を高速かつ効率的に提供．コストパフォーマンスが良く，大規模プロジェクトに適したモデル．

本研究では GPT-4.0 を用いて提案システムの動作検証を行っている．

4.2 Processing について

Processing とは，ビジュアルアートに適したプログラミング言語であり，統合開発環境である．基本的には Java と同じ構文だが，より記述がシンプルであり，視覚的なフィードバックが即座に得られるため，プログラミング初心者やプログラミングを専門としないアーティストによる制作作業によく用いられる．図 3 は Processing によって作成された画像の例である (引用元は [5])．20 行程度の分量で画像のような作品を作る事ができる．

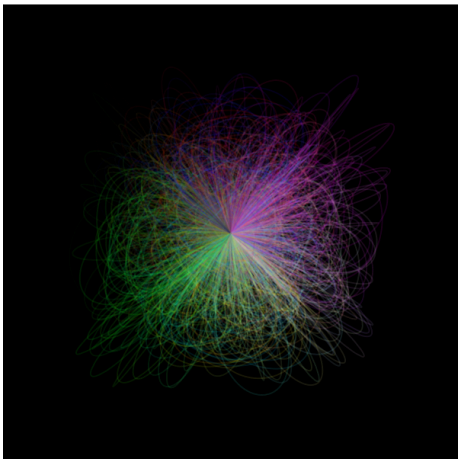

図 3 Processing によって作成された画像の例

提案システムでは p5.js[7] を用いて Processing のコードを Web ブラウザ上で実行している．

4.3 Librosa について

Librosa とは，音楽・オーディオ分析を行うための Python ライブラリ [6] であり，これを用いることでクロマグラム

やメルスペクトログラムなどの音響特徴量を抽出できる．本研究では音楽の BPM やオーディオスペクトラムを抽出するのに用いる．

4.4 本研究で用いる音響特徴量と映像表現への応用例

提案手法で用いる音響特徴量は以下の 4 種類となる．

(1) BPM
(2) ビートフレーム
(3) クロマ特徴
(4) RMS

4.4.1 BPM

Beat Per Minutes．一分間の拍数のこと．この音響信号を用いることにより，例えば音楽のリズムに合わせて一定のテンポでオブジェクトが生成されるようなアニメーションを作ることができる．

4.4.2 ビートフレーム

ビートがあるフレームを抜き出したもの．この音響信号を用いることにより，例えば音楽のビートに合わせてオブジェクトが動くようなアニメーションを作成できる．

4.4.3 クロマ特徴

12 の音階ごとの音の大きさ (またはエネルギー分布) を捉えるもの．この音響信号を用いることにより，例えば各音階のエネルギーの大きさに合わせてもののサイズが変わるようなアニメーションを作成できる．

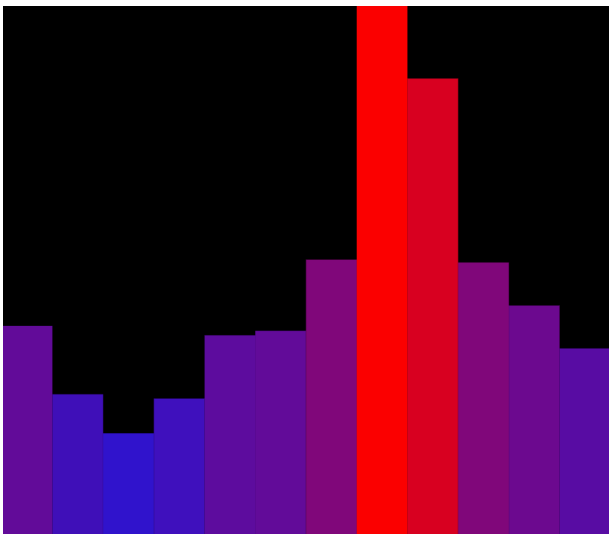

図 4　クロマ特徴を用いた映像表現の例

4.4.4 RMS

Root Means Square(二乗平均平方根)．音の持続的な強さを表す値であり，一般的に RMS は曲の盛り上がり具合と相関があるため，この音響信号を用いることで例えば音楽が盛り上がっているところで画面全体の色味やエフェクトの強度を変えるといった演出ができる．

4.5 現状の提案システムの外観

現状の提案システムの外観を図 5，6 に示す．図 5 は提案システムの入力画面，図 6 は出力画面となっている．

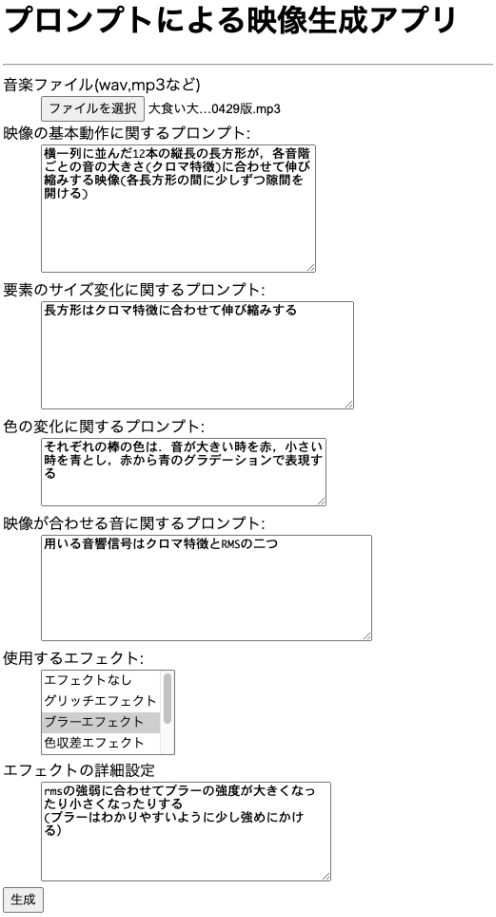

図 5　提案システムの入力画面．項目ごとに別々に入力する．

5. 評価

評価は以下の二つの比較実験によって行う．

5.1 提案システムと動画編集ソフトの，動画編集未経験者の映像制作時間の比較実験

まず，被験者 (対象は動画編集未経験者) は提案システムを用い，指定された音楽に合う映像を作成する．続いて，動画編集ソフト (AfterEffects) のレクチャーを 10 分ほど行ってから，提案システムを用いて作成した動画と似たものを作ってもらう．提案システムを使った場合に動画の作成にかかった時間と動画編集ソフトによる動画の作成にかかった時間をそれぞれ比較する．ここで作成してもらった動画はのちの実験で使用する．

5.2 提案システムと動画編集ソフトによって制作された映像のクオリティの比較実験

被験者は前述の実験の被験者に，映像制作経験のある人

```
        const duration = 32.705306122448898;
const tempo = 123.046875;
const beat_frames = Object.freeze([3,23,44,65,86,106,127,148,168,189,210,230,251,271,292,313,334,354,374,396,416,436,458,478,499,515,5
const chroma = Object.freeze([[0.82764046,0.46822733,0.67016485,0.9072268,0.58428,0.39744997,0.3773313,0.39182034,0.49551645,0.40253302,
const rms = Object.freeze([0.07250902,0.19351667,0.24351846,0.26401693,0.2690671,0.21136855,0.1609668,0.13562489,0.11179593,0.10057819,
let rectWidth, maxHeight;

function setup() {
  createCanvas(800, 600);
  rectWidth = width / 14; // Including margins between rectangles
  maxHeight = height * 0.8;
  noStroke();
}

function draw() {
  background(0);

  let currentFrame = floor((frameCount / frameRate()) * 100);
  if (currentFrame >= chroma[0].length) { // Loop the visualization if the music ends
    frameCount = 0;
    return;
  }

  // Draw rectangles based on chroma values for the current frame
  for (let i = 0; i < 12; i++) {
    let chromaValue = chroma[i][currentFrame];
    let rectHeight = chromaValue * maxHeight;

    // Map chroma value to a color between blue and red
    let colorValue = map(chromaValue, 0, 1, 255, 0);
    fill(colorValue, 0, 255 - colorValue);

    // Calculate the x position for each rectangle and leave space between them
    let x = rectWidth * i + (width - rectWidth * 12) / 2;
    rect(x, height - rectHeight, rectWidth, rectHeight);
  }

  // Apply blur effect based on RMS value
  let rmsValue = rms[currentFrame];
  let blurAmount = map(rmsValue, 0, 1, 5, 20); // Adjust the strength of blur

  applyBlur(blurAmount);
}

function applyBlur(amount) {
  push();
  filter(BLUR, amount);
  pop();
}
```

図 6 提案システムの出力画面．コードと実行結果が画面に表示される．

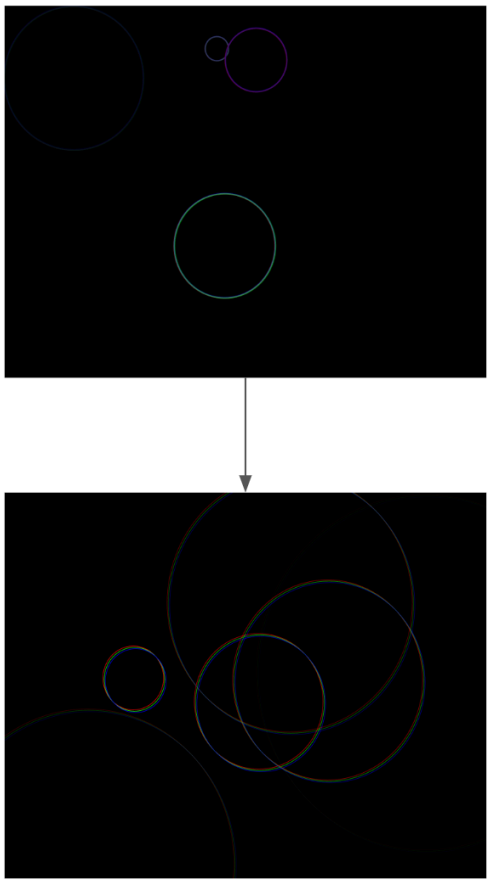

図 7 プロンプトの変更による出力変化の例

数名を加えたものとする．前述の実験で作成された動画について，提案システムで作ったものと動画編集ソフトで作ったものに，映像と音がどれだけリンクしているか，どの程度映像が好みか，といったいくつかの項目の主観的な評価を 5 段階でつけてもらう．つけてもらった評価に関してそれぞれ比較を行う．

6. 現状と今後の課題

数種類の映像表現について ChatGPT にコードを書かせてみたところ，求めている映像のイメージを細分化して指示を与えることで簡単なアニメーションであれば生成できる事がわかった．さらに，プロンプトのわずかな変更により出力のテイストを大きく変えることができることもわかった．(図 7 はエフェクトの詳細設定のプロンプトで「色収差がわかりやすいよう少し強めにエフェクトをかける」と言う一文を追加した際の出力の変化)．しかし，時間とともにアニメーションが変化していく様子や複雑なアニメーションの生成は，与える指示が長くなり過ぎてしまう，映像表現の言語化が難しくなってしまうといった問題があるため，今後生成方法やプロンプトの与え方を工夫することで改善していきたい．また，今後の改良により，入力した文字が画面上に表示される，マウスやボタンの操作によって画面の表示に変化を生むなど，よりインタラクティブな映像を生成できるようにしたり，ChatGPT のバージョンごとにコードの生成結果がどのように変わるのかを検証をしたい．さらに，ChatGPT 以外の生成 AI(Gemini や Claude など) の利用を検討している．

参考文献

[1] Sora. OpenAI. https://openai.com/sora/.(参照 2024-12-21)
[2] Vivian Liu, Tao Long Nathan Raw, Lydia Chilton. Generative Disco: Text-to-Video Generation for Music Visualization. (2023-04-17)
[3] 中渡瀬 秀一. 学術論文の情報構造認識のためのプロンプト評価. 人工知能学会全国大会論文集. (2024)
[4] Weiheng Bai, Keyang Xuan, Pengxiang Huang, Qiushi Wu, Jianing Wen, Jingjing Wu, Kangjie Lu. APILOT: Navigating Large Language Models to Generate Secure Code by Sidestepping Outdated API Pitfalls. (2024-09-25)
[5] 魚谷 知司. 作品集. Processing Fan. https://processing-fan.firebaseapp.com/galary.html.(参照 2024-09-28)
[6] Librosa. https://librosa.org/doc/latest/index.html.(参照 2024-12-24)
[7] p5.js. https://p5js.org/.(参照 2024-12-24)