漫画制作者のためのネーム制作支援システムの開発

北野 和紀 * 五十嵐 健夫 *

概要. 漫画生成を行う研究はこれまでに数多く行われてきたが, 制作プロセスに注目したものは少ない. 実際の現場では, 漫画家の知識や経験を最大限に活かしつつ, 漫画制作を効率化できるシステムが求められる. 本研究では, 漫画家が使用することを想定したネーム制作の支援システム「NameGen」を提案する. NameGen は, 漫画のシナリオを入力として受け取り, 各コマのラフスケッチと吹き出しを生成する. この生成は, 漫画シナリオをコマごとの小さなシナリオに分割した後, 各コマごとにラフスケッチを生成し, 実際の漫画のコマを参照として吹き出しを配置することで行う. 実際の漫画シナリオを入力とした実験では, NameGen が漫画家が作成したものと同じようなラフスケッチと吹き出しを生成できることを示した. さらに, 我々がこれまでに取り組んできた他の生成手法と比較することで, NameGen の有効性を確かめた.

1 はじめに

漫画は日本の文化として長年制作され続けてきた. 研究の分野でも, 映画やゲームなどの映像コンテンツを漫画に変換する研究 [16, 17, 22, 24] や, 入力画像を漫画風のレイアウトにして配置する研究 [8, 23] などが行われてきた. さらに, LLM の登場以降は, シナリオから漫画画像を生成する研究 [11, 25] も行われるようになった. しかし, これらの研究は実際の漫画の制作プロセスに注目していない. 以下に漫画会社へのヒアリングに基づく実際の漫画制作プロセスを記載する.

1. 漫画全体のシナリオを書く
2. シナリオをコマに分ける
3. コマ割りをする
4. ラフスケッチと吹き出しを描く
5. コマ割りを調整する
6. 仕上げを行って実際の漫画を描く

なお, 5 の過程では, 目線を引くために重要なコマを大きくしたり, ページ数の制約を満たすために複数のコマを一つにまとめたりする.

入力として人物やシーンの画像を与える必要がある既存研究は, シナリオから描き始める上記の漫画制作過程に当てはめることができない. LLM を用いてシナリオから漫画生成を行う研究は, シナリオとは別にレイアウトを入力する必要がある研究 [25] と吹き出しの生成ができない研究 [11] のみである. 本研究では, 上記のうちに 2 〜 4 の過程を対象として, 漫画家の制作支援のためのネーム生成システム「NameGen」を提案する. NameGen では, 漫画全体のシナリオのみを入力とし, コマごとのラフスケッチと吹き出しを生成する. 具体的には, LLM を用いて全体のシナリオをコマごとのシナリオに分割した後, ファインチューニングされた画像生成モデルによってコマごとのラフスケッチを生成する. その後, 各ラフスケッチに対して, 類似する実際の漫画コマを検索し, それをもとに吹き出しを配置する. ただし, NameGen のコマ割りは 4 コマ漫画のように正方形を縦に並べるものとした. これは, 実際の漫画制作現場でも, まずは正方形のコマを縦に並べたのち, ページ数の条件やコマの中身の重要度に合わせて大きさを調整している場合があるからだ.

実際の利用においては, 漫画家は自身が書いたシナリオを NameGen に入力することで各コマごとのラフスケッチと吹き出しの候補を得る. その中から気に入ったものを使用し, コマ割りをページ数に合わせて調整することで漫画の下書きであるネームを素早く作成する.

以下の章では, 漫画制作に関する先行研究の紹介とそれに対する本提案の位置付けを説明する. 次いで, 本提案の詳細な実装と我々が行った別の生成手法について紹介する. 実際の漫画をもとにしたいくつかの生成例を紹介した後, 現状の課題や改善の方向性を述べて, 結言とする.

2 関連研究

2.1 画像を入力とする漫画生成

画像を入力とする漫画生成は, コマ割りと吹き出しの配置を通じて行われる. なお, この節では, 映像を入力とする漫画生成も画像を入力とする漫画生成として扱っている. この種の研究は, 映像から漫画に使用する画像フレームを抽出しているため, 抽出以降の過程は画像を入力とした漫画生成として捉えることができる.

* 東京大学

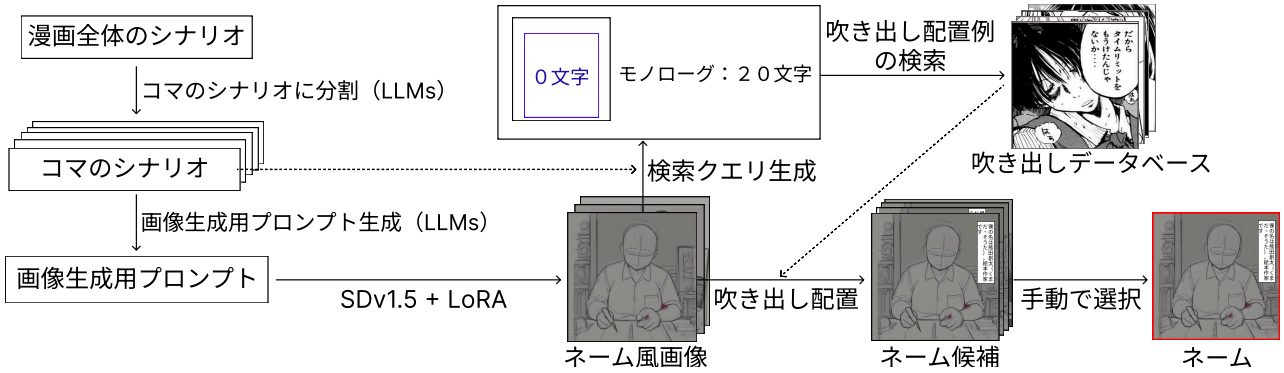

図 1. NameGen の全体像.

2.1.1 コマ割り

複数の画像から漫画風のコマ割りを生成する先行研究 [6–8, 12, 14, 16–18, 22–24] は主に, ルールやテンプレートベースのもの [6, 12, 14, 16, 18, 21–24] と, 機械学習を用いるもの [8, 17] に分けられる. ルールやテンプレートベースの手法では生成されるコマ割りの種類に限りがあったため, Cao ら [8] は初めて機械学習によるコマ割りを行った. 本研究は正方形を並べる簡易的なコマ割りとしたが, 機能の拡張としてこれらを用いることが想定される.

2.1.2 吹き出しの配置

吹き出しの配置に関する研究はわずかしか存在しない. Comic Chat [18] はコマの上側に貪欲的に吹き出しを配置したが, キャラクターと吹き出しの位置が離れてしまった. Chun ら [13] は, 吹き出し同士の距離, 及び, 話者と吹き出しの距離を考慮して吹き出しの配置を行った. しかし, 映像の漫画化を行うこの研究は, コマに含まれるセリフの量が多く, モノローグとダイアログの区別もしていないため, コマに含まれるセリフの量や種類が多様な実際の漫画の吹き出し配置とは異なる. Look Over Here [9] は, 機械学習によって読者の視線移動に基づく吹き出し配置を行った. しかし, 視線移動はキャラクターの位置とコマ割りから予測されるため, それらが全て揃っている状況でしか吹き出しの配置ができない. これらの研究とは対照的に, NameGen は, コマごとに実際のコマに近い多様な吹き出し配置を行うことができる. NameGen では, 実際の漫画のコマで構成されるデータベースに対して, 生成されたネーム風画像と似た構造を持つコマを検索し, 参照することで吹き出しの配置を行う.

2.2 テキストを入力とする漫画生成

LLM の登場以降, 漫画をテキストから生成する研究が行われるようになった [11, 25]. Chen ら [11] は, プレーンテキストを入力として漫画を生成する研究を初めて行った. コマ内とコマ間のトランスフォーマーブロックの設計により, ストーリーのテキストに基づいて 1 枚の漫画ページを生成した. しかし, 吹き出しの配置や生成はできなかった. DiffSensei [25] は, コマ割り, 登場人物と吹き出しの配置, 各登場人物の画像, 各コマの描写を入力として, 描写に沿った登場人物のバリエーションの画像を生成した. DiffSensei は入力として登場人物や吹き出しの配置が与えられることを前提としているため, 登場人物と吹き出しの配置を行う本提案とは目的が異なる.

3 方法

今回取り組むタスクは, 漫画全体のシナリオを入力として各コマごとにラフスケッチと吹き出しを生成するものである. 我々は三つの手法でこのタスクに取り組んだ. そのうち最も良い結果が得られた手法を NameGen と呼ぶことにし, 3.1 で詳細に説明する. 3.2 では, 残りの二つの手法について概要のみを簡単に説明する.

3.1 NameGen

図 1 に示すように, NameGen は, 漫画全体のシナリオを入力とし, コマごとのラフスケッチと吹き出しを生成する. 具体的には, まず, LLM を用いて全体のシナリオをコマごとのシナリオに分割 (3.1.1) した後, コマごとの画像生成用プロンプトを生成する (3.1.2). 次に, Stable Diffusion [3] とラフスケッチ画像でファインチューニングされた LoRA [2] を用いて, 前述の画像生成用プロンプトからラフスケッチを x 枚生成する. その後, 事前に構築した吹き出しデータベース (3.1.3) に対して, 各ラフスケッチごとに作成された検索クエリ (3.1.4) を用いて, 類似する構造の漫画コマを k 件検索する (3.1.5). 検索の結果得られた漫画コマを参考として, ラフスケッチに吹き出しを配置する (3.1.6). 最終的に得られる $x \times k$ 枚の候補から人間が最も良いと感じたものを選択して最終結果とする.

3.1.1 シナリオの分割

漫画は主に複数のコマによって構成される．しかし, 実際の漫画制作現場と同様, NameGen ではコマごとに分かれていない全体のシナリオのみが与えらえる．したがって, 最初に漫画全体のシナリオをコマごとのシナリオに分割する必要がある．

実際の漫画のシナリオ分割の観察をもとに, LLM に下記に述べる条件を与えてシナリオを分割する．

- 場面が変わる描写やセリフは同じコマに含めない
- 一つのコマに含まれる行動は一つのみとする
- 一つのコマに含まれる会話は一往復までとする
- 一つのコマに含まれる同じ人物のセリフは一つまでとする

さらに, セリフを表す文については, 話者抽出, 及び, モノローグかダイアログかの判定を行った．

なお, ここで分割された各コマのシナリオは必ず元の文章から構成される．LLM による文章の生成や要約は行われていない．

3.1.2 画像生成用プロンプトの生成

本項では, 画像生成モデルに渡すラフスケッチ生成用のプロンプト生成について述べる．プロンプト生成では, LLM を 3 つのステップで使用する．まず最初のステップでは, コマのシナリオに含まれる登場人物のローマ字読みを取得する．これによって, 固有名詞をそのまま与えることによる別の名詞との混同を避ける: 固有名詞の「蝶子」-昆虫の「蝶」, 固有名詞の「カエデ」-植物の「カエデ」．次のステップでは, 最初のステップで得られた固有名詞とローマ字読みの関係を与えた上で, 複数の文章（セリフの文章と描写の文章）を一つの文章にまとめる作業を行う．最後のステップでは, 以前のステップで一つにまとめた文章を画像生成用の詳細な文章に変換する．ここで生成したプロンプトは画像生成モデル (Stable Diffusion v1.5 + LoRA) に与えられ, ラフスケッチの生成に活用される．

3.1.3 吹き出しのデータベース構築

NameGen では事前に構築したデータベースに対して, ラフスケッチと似た構造を持つ漫画コマを検索して参照することで吹き出しの配置を行う．本項では, 高速な検索を可能にする吹き出しのデータベースの構築方法について述べる．

吹き出しのデータベース構築には Manga109 [5, 15,20], Manga109Dialog [19] を用いる．Manga109 は 109 冊の漫画について, 1 ページごとに画像とアノテーションデータを記したデータセットである．Manga109Dialog は, Manga109 についての追加のアノテーションデータであり, 各吹き出しに対して話者の対応づけをしている．

まず, Manga109 のアノテーションデータをもとにアノテーションと画像を各コマごとのものに分割する．この時, 複数のコマに跨っている吹き出しは除外する．その後, Manga109Dialog をもとにキャラクターの分類と吹き出しの分類を行う. キャラクターについては, 吹き出しと対応づけられている場合は話者, そうでない場合は非話者として分類する. 吹き出しについては, 対応する話者がコマ内に存在する場合はダイアログ, そうでない場合はモノローグと分類する．この分類方法では, キャラクターの心情を表す吹き出しとコマにいないキャラクターによる吹き出しを区別できない．本来モノローグは前者のみを表すが, 吹き出し配置の参考として使用する際には, コマに話者がいない吹き出しという点で同一視できる．

最後に, 高速な検索を可能にするためにインデックス付けを行う. 具体的には, コマに含まれる非話者数, 話者数の組み合わせごとにデータベースをいくつかのグループに分割する．

3.1.4 吹き出し配置のためのクエリ生成

本項では, 吹き出しのデータベースからラフスケッチと類似するコマを検索する際に使用する検索クエリの生成方法について述べる．

検索クエリはモノローグ文字数, レイアウトという 2 つの要素によって構成される．モノローグ文字数は話者をコマ中に持たないセリフの総文字数を表す．NameGen では, コマごとのシナリオにおけるモノローグのセリフの文字数を合計することで取得する．レイアウトは, コマ内の人物の位置とその人物のセリフの文字数を記載したものである．レイアウトを得るために, まず, ラフスケッチに Openpose [10] を適用して人物の位置を得る．その後, 右に位置するものから順番にコマごとのシナリオにおけるダイアログのセリフを割り当てる．「得られた人物の数 $>$ そのコマのダイアログのセリフ数」の場合は, セリフを割り当てられなかった人物は非話者として位置だけを記載する．なお,「得られた人物の数 $<$ ダイアログのセリフ数」の場合は, 以降の処理は行わずに中断する．

3.1.5 吹き出し配置の検索

本項では, 検索クエリから吹き出し配置を検索する方法について述べる．

まずは, 条件をもとに簡単な絞り込みを行う．吹き出しのデータベース内のデータは, 3.1.3 で述べたように話者数と非話者数の組み合わせでグループ分けされている．まずは, 検索クエリの話者数と非話者数の組み合わせをもとに候補を絞り込む．その後, 検索クエリのモノローグ文字数と候補レイアウトのモノローグ文字数を比較して, 一定以上離れているものは除外する．

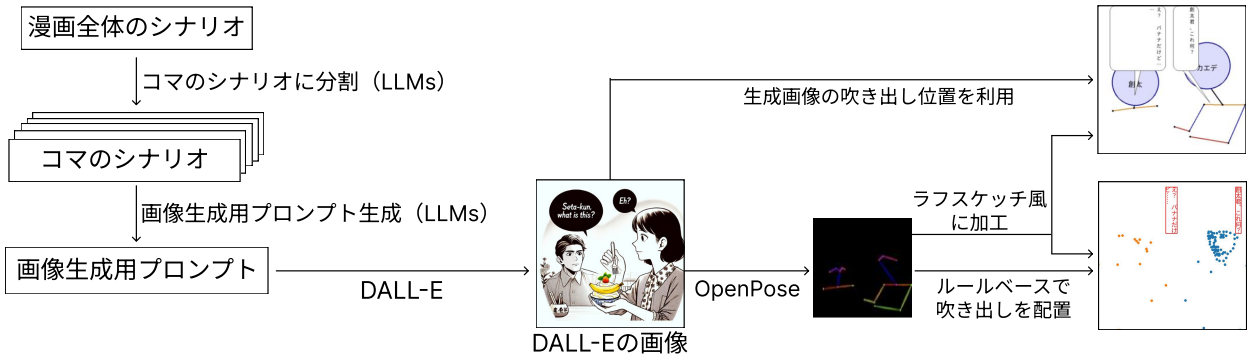

図 2. ネーム生成の他の手法 (生成画像の吹き出しの位置を利用する手法とルールベースで吹き出しの位置を決定する手法.)

次に, 上記絞り込みによって残った全ての候補に対して類似度を計算する. 類似度の計算は, 検索クエリのレイアウトと候補のレイアウト間の重み付き二部グラフの最適マッチングによって行う.

レイアウト中の話者と非話者をレイアウト要素として定義し, e または $\hat{e}$ を用いて定義する. 検索クエリのレイアウト $l = \{e_1, e_2, ..., e_m\}$ と候補レイアウト $l' = \{\hat{e_1}, \hat{e_2}, ..., \hat{e_n}\}$ を考える.

類似度は以下の最適化問題を解くことで得られる.

$$\max \sum_{i=1}^{m}\sum_{j=1}^{n} w(e_i, \hat{e}_j)\cdot\gamma_{ij}$$

$$\text{s.t.}\quad \gamma_{ij}\in\{0,1\}, \forall i\in\{1,2,\dots,m\}, \forall j\in\{1,2,\dots,n\},$$

$$\sum_{j=1}^{n}\gamma_{ij}\le 1,\ \forall i\in\{1,2,\dots,m\},$$

$$\sum_{i=1}^{m}\gamma_{ij}\le 1,\ \forall j\in\{1,2,\dots,n\}. \qquad (1)$$

ただし, $\gamma_{ij} \in \{0,1\}$ は, 要素のペア $(e_i, \hat{e_j})$ がマッチングされる時に 1 となり, それ以外の場合は 0 となる.

e_i, $\hat{e_j}$ 間の重み $w(e_i, \hat{e}_j)$ は次のように定義する.

$$w(e_i,\hat{e}_j)=\begin{cases} k\cdot \mathrm{IoU}(e_i,\hat{e}_j)+(1-k)\cdot G(t_i,\hat{t}_j), \\ \quad \text{if}\quad T(e_i)=T(\hat{e}_j)=\text{Speaker}, \\ \mathrm{IoU}(e_i,\hat{e}_j), \\ \quad \text{if}\quad T(e_i)=T(\hat{e}_j)=\text{NonSpeaker}, \\ 0, \quad \text{otherwise} \end{cases}$$

ただし, $T(e_i)$ と $T(\hat{e_j})$ はそれぞれ, e_i, $\hat{e_j}$ の属性を表し, 要素が話者の場合は Speaker, 非話者の場合は NonSpeaker となる. 要素の属性が Speaker のとき, t_i と $\hat{t_j}$ はそれぞれ, e_i, $\hat{e_j}$ が持つセリフの長さを表す. また, G は以下の式で表されるガウス関数である. 2 つの Speaker 要素が持つセリフの長さが近いほど高い値を取る.

$$G(t_i,\hat{t}_j)=\exp\left(-\frac{(t_i-\hat{t}_j)^2}{2\sigma^2}\right)$$

ただし, $k \in [0,1]$ は, IoU と G の線型結合の割合を調整するためのパラメータである.

この最適化問題をハンガリアン法を用いて解くことで類似度を計算する.

3.1.6 吹き出しの配置

ラフスケッチの吹き出しの配置は, 前項で検索した実際の漫画コマの吹き出し配置を参照して行う. 検索の結果得られた漫画コマは複数の吹き出しを含むことがある. ラフスケッチの吹き出しを配置する際に参照するものを参照吹き出しとして, 以下では参照吹き出しを選択する方法を述べる.

参照吹き出しの選択方法は, セリフがモノローグの場合とダイアログの場合で異なる. セリフがモノローグの場合は, 検索結果の漫画コマのモノローグの吹き出しのうち最も右側に位置するものを選択する. セリフがダイアログの場合は, 次に示すアルゴリズムによって選択する.

参照吹き出しを選択した後, 参照吹き出しの右上を開始点として縦書きで文字を配置する. この時, 縦幅は参照吹き出しと同じにし, 横幅を文字数に合わせて調整する. その後, モノローグ要素の場合は長方形, ダイアログ要素の場合は楕円で文字全体を囲む.

3.2 ネーム生成の他手法

ここでは, NameGen に至るまでに行った二つの別手法について説明する. 図 2 に概要を示す. どちらの手法も NameGen と同じように画像生成用プロンプトを生成し, それに基づいて画像生成を行うが, 画像生成では Stable Diffusion ではなく OpenAI の DALL-E を使用した. その後, OpenPose によって

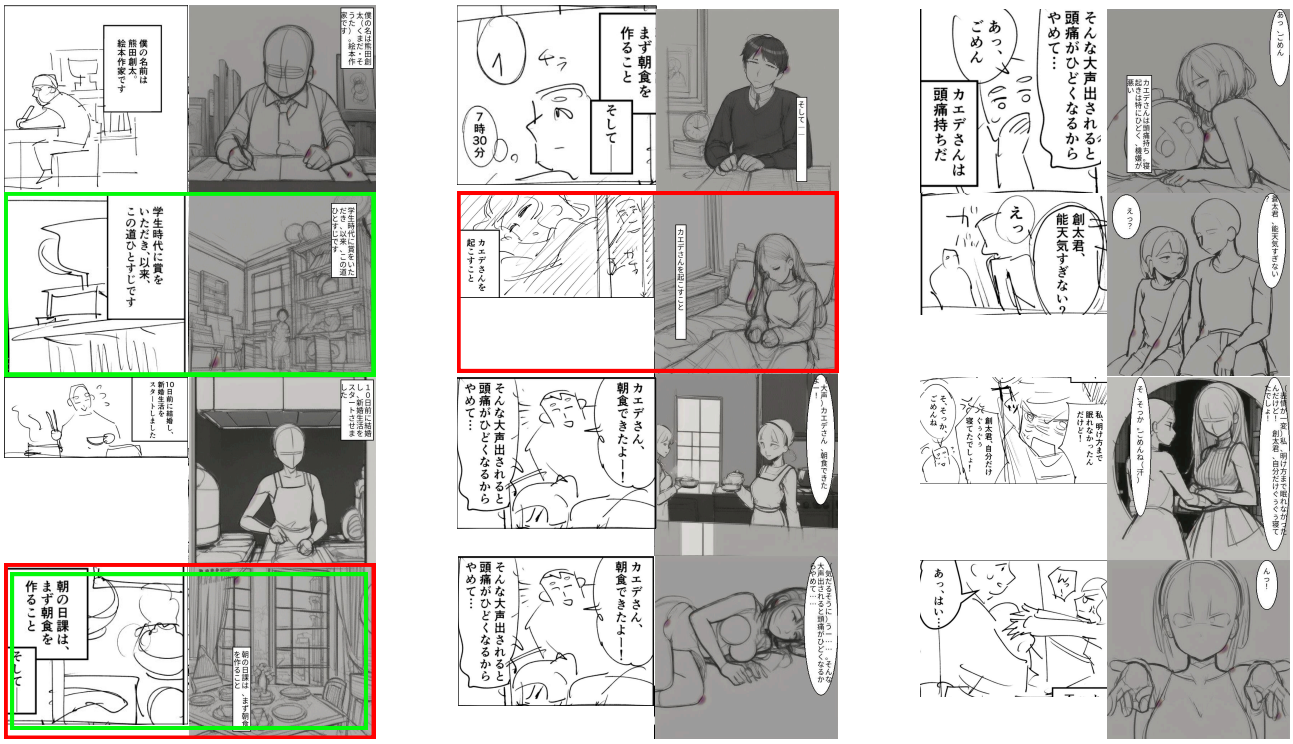

図 3. 『モラハラ DV 妻から命からがら逃げました』の実際の原稿をもとに生成した結果の例．各コマごとに，左の画像が漫画家が書いたネーム，右の画像が生成されたラフスケッチと吹き出し．赤枠は吹き出しを左側に配置できた例．緑枠は人物がいないコマを生成できた例

Algorithm 1 ダイアログ要素の参照吹き出しを決定するアルゴリズム

入力：検索結果の漫画コマ l', コマの原稿 $panel$
l' の要素 $\hat{e_1}, \hat{e_2}, \ldots$ を並びかえる．話者要素は非話者要素よりも前に来る．話者要素同士は関連づけられている吹き出しが右にあるものほど前に来る．
$poiner$ 変数を 0 として初期化.
for $\hat{e} \in l'$ **do**
  while $panel[pointer]$ がダイアログではない **do**
    $pointer \leftarrow pointer + 1$
  end while
  if $poiner \geq len(panel)$ **then**
    処理を終了
  end if
  $bbox \leftarrow \hat{e}$が持つ最右の吹き出し
  $bbox$ を $panel[pointer]$ の参照吹き出しに選択
end for=0

人物のポーズを抽出し，一つ目の手法では生成画像に含まれる吹き出しの位置，二つ目の手法ではルールベースで決定した位置に基づいて吹き出しの配置を行う．

一つ目の手法では，画像生成モデルが生成する画像にすでに含まれている吹き出しを利用する．ファインチューニングを Manga109 で行った物体検出モデル [4] を用いて吹き出しを検出する．物体検出の信頼度が高い吹き出しのうち，最も右側に位置するものから順番に吹き出しの位置の参照として利用する．生成画像はラフスケッチではないので，OpenPose によって抽出した人間の姿勢情報をもとにラフスケッチに近くなるように加工する．

二つ目の手法では，吹き出しを右上から左側に向かって順番に配置するように試みる．その過程で吹き出しが人物と重なった場合は，人物を縮小して再度配置を試みる．

これらの方法で生成した結果と NameGen の結果の比較は 4.3 に記載した。

4　実験と結果

本システムを使用して『モラハラ DV 妻から命からがら逃げました』[1]の実際の原稿をもとにラフスケッチと吹き出しを生成した．また，NameGen と 3.2 で述べた他の手法との比較も行った．

[1] © 創太・ミカイ／パルソラ

図 4. ネームの生成がうまくいかない例

4.1 実装と手順

本システムの実装で使用する LLM は全て OpenAI の gpt-4o を用いた.

NameGen では, 画像生成に Stable Diffusion v1.5 を使用した. チェックポイントは T-anime-v4 [1], LoRA は sketch anime pose [2] を採用した. 吹き出し配置の検索で使用するガウス関数 G のパラメータ σ は 10, IoU と G の線型結合の割合を調整するパラメータ k は 0.4 とした. コマのシナリオへの分割, 画像生成用プロンプトの生成はそれぞれ 1 回のみ行い, Stable Diffusion と LoRA を用いた画像生成は 3 回行った. 生成された各画像に対して吹き出し配置の検索により類似度の高い順に 4 つの参考コマを取得した. そして, それぞれの参考コマごとに参考コマをもとにした吹き出し生成を行った. 最後に, 生成された 1 2 個（3 つのラフスケッチ × 4 つの参考コマ）の結果の中から我々が最も良いと考えるものを 1 つ選び, 生成結果とした.

4.2 実際の原稿からの生成

漫画家が書いたネームと生成された結果の比較を図 3 に示す. 各列の左側の画像が漫画家が書いたネーム, 右側の画像が生成されたラフスケッチと吹き出しである. なお, 漫画家が書いたネームは, 生成結果の吹き出しのセリフをもとに対応すると考えられるものを選択した. シナリオの分割では, 図 3 の吹き出しの中身に注目すると, 二列目の下二コマ以外は実際の漫画と同じように分けることができた. ラフスケッチと吹き出しの生成では, 赤枠で囲ったコマのように吹き出しが左に位置するコマの生成も行うことができた. また, 緑枠で囲ったコマのように人物がいないコマに対してもラフなスケッチを提供することができた. 一方で, コマを通して人物の一貫性が保たれなかった. また, 図 4 に示すように, ハグをしてもらっているべき場面で単一の人物のみが生成されたり, 長いモノローグが人物に重なってしまったりするなどの問題も確認された.

4.3 NameGen と他手法の比較

図 5 に手法ごとの結果の比較を載せる. それぞれで 5 枚ずつ生成したもののうち最も良いものを抜粋した. 生成画像の吹き出し位置を利用する手法では, 吹き出しが顔に被ったり, 不自然な配置になったりする不備が多く見られた. ルールベースで吹き出しを配置する手法と NameGen はどちらも似たような吹き出しの配置になった. しかし, ルールベースの手法は, どのコマでも似たような吹き出しの配置になってしまう問題があった. NameGen ではルールベースの手法で可能な標準的な吹き出しの配置だけではなく, 図 3 で示したように多様な種類の吹き出しの配置ができた.

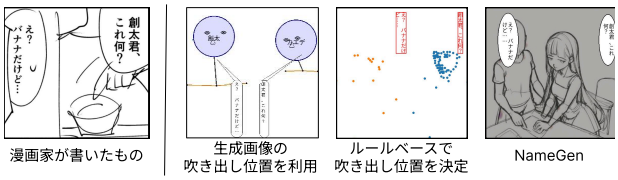

図 5. それぞれの手法ごとの結果の比較

5 課題と改善の方向性

NameGen は漫画シナリオのみから多様なパターンのラフスケッチや吹き出しを生成できる一方で課題も存在する. ここでは特に重要と考えられる三つを述べる. 一つ目は, コマ割りで正方形のコマを縦に並べるのみとしている点である. これは, Automatic Stylistic Layout [8] などの先行研究を用いることで克服できるだろう. 二つ目は, 生成結果の中から最も良いものを人の手によって選んでいるという点である. これは, 人物と吹き出しが被っているものを除外するなどのルールベースによるフィルタや, LLM に判断基準を与えて生成画像を評価するなどのランク付けによって改善できると考える. 三つ目は, セリフの話者と人物の特徴が一致しない点である. 例えば, 女性のセリフにも関わらず話者が男性である場合などがこれに該当する. 生成結果の画像に対して物体検出を行い, どの登場人物か判定した上でセリフを割り当てるなどの工夫が考えられる.

6 まとめ

本研究では, 漫画製作者の支援を目的としてネームの制作支援システム NameGen を提案した. このシステムでは, 漫画全体のシナリオをコマごとに分割し, 画像生成モデルや実際の漫画コマに基づく吹き出し配置によって, 様々なバリエーションを持つラフスケッチと吹き出しの生成を可能にした. 将来的には, コマ割りや人間が選ぶ手間の削減を行う予定である.

謝辞

図中の漫画画像は Manga109 [15] の画像を一部使用した (© 竜正, 浅月 舞, 島田 ひろかず, 花田 朔生, 記伊 孝).

参考文献

[1] Civitai T-Anime. https://civitai.com/models/69552/t-anime-v4-pruned.

[2] Sketch Anime Pose. https://civitai.com/models/106609/sketch-anime-pose.

[3] Stable Diffusion v1.5. https://huggingface.co/stable-diffusion-v1-5/stable-diffusion-v1-5.

[4] YOLO. https://github.com/ultralytics/ultralytics.

[5] K. Aizawa, A. Fujimoto, A. Otsubo, T. Ogawa, Y. Matsui, K. Tsubota, and H. Ikuta. Building a Manga Dataset "Manga109" with Annotations for Multimedia Applications. *IEEE MultiMedia*, 27(2):8–18, 2020.

[6] J. Boreczky, A. Girgensohn, G. Golovchinsky, and S. Uchihashi. An interactive comic book presentation for exploring video. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '00, p. 185–192, New York, NY, USA, 2000. Association for Computing Machinery.

[7] J. Calic, D. P. Gibson, and N. W. Campbell. Efficient Layout of Comic-Like Video Summaries. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(7):931–936, 2007.

[8] Y. Cao, A. B. Chan, and R. W. H. Lau. Automatic stylistic manga layout. *ACM Trans. Graph.*, 31(6), Nov. 2012.

[9] Y. Cao, R. W. H. Lau, and A. B. Chan. Look over here: attention-directing composition of manga elements. *ACM Trans. Graph.*, 33(4), July 2014.

[10] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh. OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields, 2019.

[11] S. Chen, D. Li, Z. Bao, Y. Zhou, L. Tan, Y. Zhong, and Z. Zhao. Manga Generation via Layout-controllable Diffusion, 2024.

[12] W.-T. Chu, C.-H. Yu, and H.-H. Wang. Optimized Comics-Based Storytelling for Temporal Image Sequences. *IEEE Transactions on Multimedia*, 17(2):201–215, 2015.

[13] B.-K. Chun, D.-S. Ryu, W.-I. Hwang, and H.-G. Cho. An Automated Procedure for Word Balloon Placement in Cinema Comics. In G. Bebis, R. Boyle, B. Parvin, D. Koracin, P. Remagnino, A. Nefian, G. Meenakshisundaram, V. Pascucci, J. Zara, J. Molineros, H. Theisel, and T. Malzbender eds., *Advances in Visual Computing*, pp. 576–585, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[14] A. Durrant, D. Rowland, D. S. Kirk, S. Benford, J. E. Fischer, and D. McAuley. Automics: souvenir generating photoware for theme parks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, p. 1767–1776, New York, NY, USA, 2011. Association for Computing Machinery.

[15] A. Fujimoto, T. Ogawa, K. Yamamoto, Y. Matsui, T. Yamasaki, and K. Aizawa. Manga109 dataset and creation of metadata. In *Proceedings of the 1st International Workshop on CoMics ANalysis, Processing and Understanding*, MANPU '16, New York, NY, USA, 2016. Association for Computing Machinery.

[16] L. Herranz, J. Calic, J. M. Martinez, and M. Mrak. Scalable Comic-Like Video Summaries and Layout Disturbance. *IEEE Transactions on Multimedia*, 14(4):1290–1297, 2012.

[17] G. Jing, Y. Hu, Y. Guo, Y. Yu, and W. Wang. Content-Aware Video2Comics With Manga-Style Layout. *IEEE Transactions on Multimedia*, 17(12):2122–2133, 2015.

[18] D. Kurlander, T. Skelly, and D. Salesin. Comic Chat. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, p. 225–236, New York, NY, USA, 1996. Association for Computing Machinery.

[19] Y. Li, K. Aizawa, and Y. Matsui. Manga109Dialog: A Large-scale Dialogue Dataset for Comics Speaker Detection, 2024.

[20] Y. Matsui, K. Ito, Y. Aramaki, A. Fujimoto, T. Ogawa, T. Yamasaki, and K. Aizawa. Sketch-based Manga Retrieval using Manga109 Dataset. *Multimedia Tools and Applications*, 76(20):21811–21838, 2017.

[21] D.-S. Ryu, S.-H. Park, J.-w. Lee, D.-H. Lee, and H.-G. Cho. CINETOON: A Semi-automated System for Rendering Black/White Comic Books from Video Streams. In *2008 IEEE 8th International Conference on Computer and Information Technology Workshops*, pp. 336–341, 2008.

[22] A. Shamir, M. Rubinstein, and T. Levinboim. Generating comics from 3D interactive computer graphics. *IEEE Computer Graphics and Applications*, 26(3):53–61, 2006.

[23] H. Tobita. Comic Engine: interactive system for creating and browsing comic books with attention cuing. In *Proceedings of the International Conference on Advanced Visual Interfaces*, AVI '10, p. 281–288, New York, NY, USA, 2010. Association for Computing Machinery.

[24] M. Wang, R. Hong, X.-T. Yuan, S. Yan, and T.-S. Chua. Movie2Comics: Towards a Lively Video Content Presentation. *IEEE Transactions on Multimedia*, 14(3):858–870, 2012.

[25] J. Wu, C. Tang, J. Wang, Y. Zeng, X. Li, and Y. Tong. DiffSensei: Bridging Multi-Modal LLMs and Diffusion Models for Customized Manga Generation, 2025.