テキストプロンプトによるベクター形式ロゴ画像生成

山倉 隆太[†1,a)]　柳井 啓司[†1,b)]

概要：本研究では，近年提案された微分可能レンダラー [9] とラスター画像生成における強力な性能を有する Stable Diffusion [14] を用いることで，テキストプロンプトと形状の画像から算出される損失を用いてベクターパラメータの最適化を行い，ベクター形式ロゴ画像生成を行う．また，ベジェ曲線の最適化に伴う自己交差問題 [3] に対して，ベジェ曲線セグメント間を考慮した Radiation Loss を新たに提案し導入した．本手法により入力テキスト及び形状を維持したロゴ画像を生成することが可能である事が示された一方で，不要なパスが残留することやテキストによる制御が難しいことなど，今後の改善点も明らかとなった．

キーワード：画像生成，ベクターグラフィックス，微分可能レンダラー

1. はじめに

画像をパラメトリックな数学的プリミティブにより表現するベクター画像は，拡大・縮小の操作に画質の劣化を伴わない点や，多くの場合に同等の解像度を持つラスター画像よりもデータサイズがコンパクトである等の特有の利点から，フォントやロゴなどの多くのグラフィックデザインで標準的に用いられている．一方，その作成には必要な文法や描画ツールの習熟などが必要であり，多大な労力と時間を必要とすることから，テキストや画像を入力とした柔軟かつ直感的なベクター画像の生成は芸術活動の支援につながるものと考えられる．本論文では特にロゴ画像を目標として，入力画像の形状を維持した，テキストの内容を反映する多様なベクター画像の生成を行う．

現在，生成モデルによるベクターグラフィックスの生成は主に，学習時に SVG などのベクターグラフィックスに特有のコマンドシーケンスに対する明示的な監視を必要とする生成モデルを使用する言語ベースの方法と，事前学習済みのラスター画像生成モデルと画像のベクトル化を組み合わせた画像ベースの方法がある．このうち前者は，大規模で高品質のベクター画像の収集が困難であることや，ベクター画像の文法表現が一意でない場合があることから，多くの場合ロゴやフォントなどの特定のドメインに表現が制限される．これに対し，後者は大量のラスター画像により学習された生成モデルの知識をそのまま活用可能であり近年の堅牢な画像生成モデルの恩恵を受けることができる．加えて，微分可能レンダラー [9] の登場により，複雑なベクトル変換セクションを考慮せずに，損失を用いた単純な最適化プロセスを経てラスター画像のベクトル化が可能となった．本論文においても後者の方法を用いて Text-to-Image モデルの知識を活用することで，多様かつ安定したベクターグラフィックスの生成を目指す．具体的には，入力テキストプロンプトから Stable Diffution の知識蒸留プロセスを経て目的画像のコンテンツを入手しつつ，形状を示す画像による制限の元で，特にロゴ画像を目的とした T2I ベクター画像生成を行う．

[†1] 現在，電気通信大学
Presently with The University of Electro-Communications
[a)] yamakura-r@mm.inf.uec.ac.jp
[b)] yanai@cs.uec.ac.jp

2. 関連研究

2.1 微分可能レンダラー

従来，ベクターグラフィックスのレンダリング工程は一方向であり，ラスター画像のベクタライズにはエッジのトレースを伴う特殊なメソッド [5], [18] が必要であった．しかし，これらの手法によるベクタライズは元のベクトルメトリクスと無関係であり生成されるベクターグラフィックスは劇的に異なる構造を持つため，ラスターベースのアルゴリズムをベクターグラフィックスに適用することができない．Li ら [9] はこのようなベクタライズ手法の問題点に対して微分可能な 3D レンダラー [10] に基づき，ピクセルの事前フィルタリングに関して，分析プレフィルターとマ

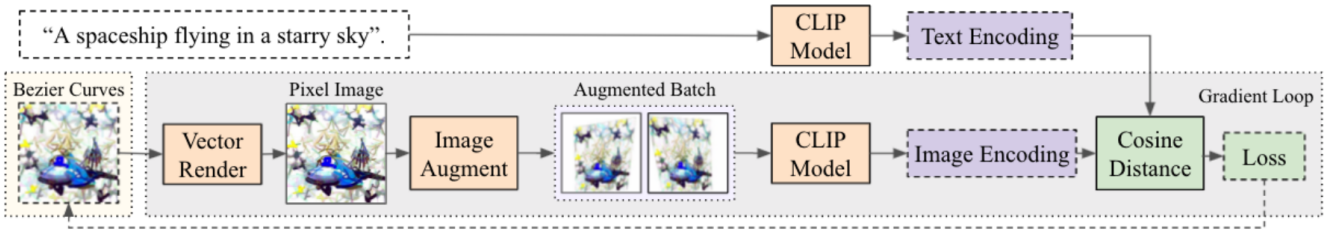

図 1 CLIPDraw のフレームワーク ([3] より引用)

ルチサンプリングアンチエイリアシングの 2 つの手法を用いることで入力ベクトルパラメータに対して勾配を自動的に計算できるレンダラーを提案した．

2.2 微分可能レンダラーによる画像生成

微分可能レンダラーの登場によって，高品質かつ大規模なベクターコンテンツのデータ収集の必要性の制限を回避して，ラスターベースの損失関数や機械学習手法をベクターコンテンツへ直接応用してベクターパラメータを最適化するいくつかの手法が提案された．

CLIPDraw [3] は，OpenAI により発表された画像とテキストの関係性を学習した大規模なモデル CLIP [13] を用いて，入力テキストと出力画像の類似度を計算し，微分可能レンダラーを経てベジェ曲線のパラメータを最適化する手法であり，以後の同様のフレームワークのベースとなった．図 1 に CLIPDraw のフレームワークを示す．さらに，StyleCLIPDraw [17] では VGG16 モデルを使用して補助スタイル損失を伴う画像を条件付けすることで CLIPDraw で生成された画像にスタイルを転送する拡張を行った．ただし，これらの研究はベクターパスによる描画を主たる目的としているため，使用されるベクタープリミティブが曲線に限定され表現が稚拙になる点が課題として残る．

VectorFusion [8] では，学習済みのテキストからの画像生成（T2I）モデルである Stable Diffusion [14] を用いて，拡散モデルの出力を 3D やベクターパラメータなどの任意のパラメータ空間に変換するようにモデルを蒸留するメソッド Score Distillation Sampling (SDS) [12] をベクターグラフィックス向けに調整して損失関数として利用することによって，より一貫したベクターグラフィックスの生成を可能とした．なお，本研究においても VectorFusion で用いられた SDS Loss を用いる．図 2 は VectorFusion による生成例である．VectorFusion は CLIPDraw に比べ，拡散モデルの頑強な出力空間を用いたことで大幅に描画能力が向上したが，入力としてテキストのみを受け取るため入力画像の形状を維持する機能が存在しない．

また、本研究の目的の一つである入力画像の形状の制限

図 2 VectorFusion の生成例 ([8] より引用)

を受けるという制約は，形状の意味が重要になるフォント生成タスクで多く見られる．

Zero-shot-font [7] は，CLIPStyler [4] で提案された CLIP Loss の拡張である Patch CLIP Loss と directional CLIP Loss，およびコンテンツ画像の距離変換画像を用いた Distance Transform Loss [1] を用いることで，フォント画像に対する高品質なスタイル変換を行った研究である．特に Zero-shot-font では本研究と同様に単純なアイコン画像を入力とした場合のスタイル変換が可能である事が示されているが，この点に関して本研究はスタイルのような局所的な構造ではなく入力画像の形状を維持した大域的な構造の生成を目指すという点で異なる．

3. 提案手法

3.1 手法概要

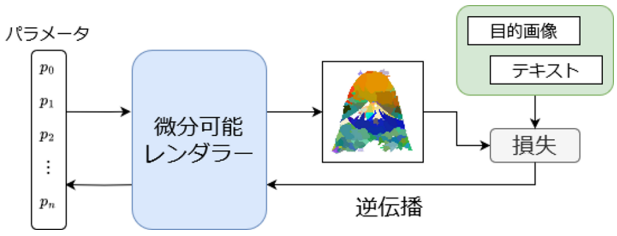

図 3 手法概略図

手法概要図を図 3 に示す．本手法でも，CLIPDraw [3] のアイデアに基づき，入力画像 I からランダムに初期化されたベジェ曲線パラメータ（色，不透明度及び制御点座標）p_i を微分可能レンダラーによって勾配発生可能な状態でレンダリングを行い，ラスター画像を生成する．出力のラス

ター画像と，入力プロンプト及び形状画像を使用して，3.4 章に示す損失関数から損失を取得して逆伝播を行い，勾配降下法によってベクターパラメータの最適化を行うことで，目的の出力を得る．

3.2 表現形式と初期化

ベクター画像を構成するプリミティブは，CLIPDraw のアイデアに基き，閉じた始点 3 次ベジェ曲線に限定する．ベジェ曲線は制御点群により定義される曲線であり，ベジェ曲線同士を連結して用いることで多様な形状に対して近似が可能であり，表現力を損なわずにシンプルな実装と評価を行うことができる．このとき，N 本の閉じたベジェ曲線パスを用いてベクターグラフィック $\hat{I} = \{P_0, ..., P_{N-1}\}$ を形成するものする．各パスは M 本の 3 次ベジェ曲線セグメントによって構成され，セグメントを構成する 4 つの制御点 p_m により $P_n = \{p_{0,0}, p_{0,1}, p_{0,2}, ..., p_{M-1,0}, p_{M-1,1}, p_{M-1,2}\}$ と定義される．なお，パスは必ずセグメントの始点制御点を通過し，セグメントの終点制御点は接続のために次のセグメントの始点制御点と共有される．また，各 P_n はそれぞれ単一の不透明度を含む色を保有し，制御点座標及び色を個別に最適化することでベクター生成を行う．

ベクターパスの初期化は LIVE [11] のアイデアに従って，すべての制御点を円形に配置することで行う．これによりパスの自己交差問題を事前に防ぐ効果が期待される．さらに Tone Loss の収束速度のためにあらかじめ入力画像 I の範囲内にパスの中心座標を設定する．図 4 に初期化の例を示す．

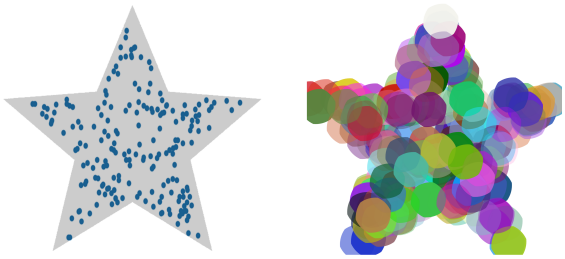

図 4 初期化の例．入力画像に従ってベジェ曲線の中心位置をランダムに取得し (左図)，円形に初期化を行う (右図)．

3.3 自己交差問題

Ma ら [11] は，パスの初期化や最適化プロセスの結果，最適化の結果ベクターパスが自己交差することで有害なアーティファクトの生成や不適切なトポロジーにつながることを指摘している．図 5 左上に示されるように自己交差問題が発生したパスは既定のレンダリングサイズを超える拡大処理を行った際にアーティファクトが発生することや，アーティファクトをカバーするための追加のパスが発生する可能性がある．同研究ではこの問題に対応するために，すべてのベクタープリミティブが 3 次ベジェ曲線であるという仮定のもと，あるパスの制御点を順番に A, B, C, D とした場合に，$\overrightarrow{AB}$ と $\overrightarrow{CD}$ のなす角が 180 度を超えないように調整する Xing Loss を提案した．ただし，Xing Loss では接続されたベジェ曲線セグメント同士の交差を防ぐことができないため，ターゲット画像に直接最適化されない本研究においては不十分であると考える．この点に関して，3.4.3 章にて，各セグメント間を含めて自己交差を解消する Radiation Loss を提案して導入する．

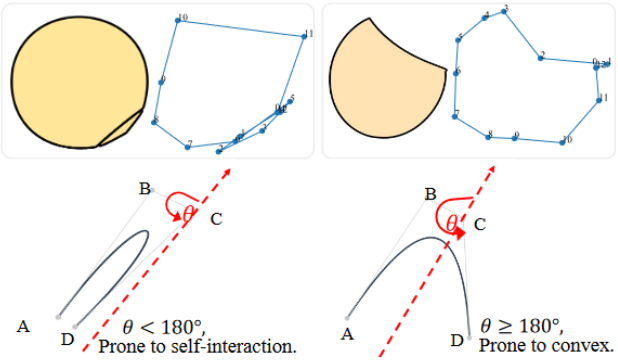

図 5 自己交差問題および Xing Loss の概要図（[11] より引用）

3.4 損失関数

図 6 に損失関数の導出フローを示す．損失関数には入力画像の領域に形状を制限する Tone Preserving Loss，入力テキストのコンテンツを反映する SDS Loss，ベクターパスの品質を保つ Radiation Loss を導入する．

3.4.1 Tone Preserving Loss

Tone Preserving Loss は Word-As-Image [6] で提案された損失であり，レンダリングされたベクター画像と参照する形状の画像にローパスフィルタを適用し，画像間のユークリッド距離をとる．画像の局所的なトーンを保存することによって，調整された画像が入力画像から大きく逸脱しないように制限する．

$$\mathcal{L}_{tone} = ||LPF(\mathcal{R}(I)) - LPF(\mathcal{R}(\hat{I}_i))||_2^2 \quad (1)$$

ここで，$\mathcal{R}(\cdot)$ はレンダラーであり，P および $\hat{P}$ はベクター画像を構成する一連のパラメータである．

本手法でも入力画像の形に形状を制限するために $\mathcal{L}_{tone}$ を導入する．ただし，Word-As-Image はモノトーンのフォ

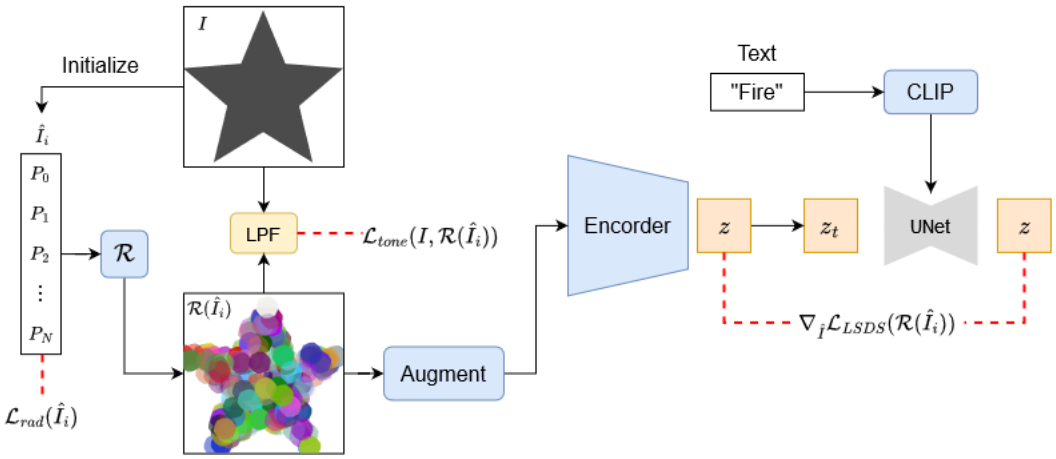

図 6 損失関数の概要

ント形状操作を目的とした手法であるため式 1 を直接適用またはレンダリングされた画像にグレースケール変換を適用した結果に適用すると，色価がマスクの値に近づくように最適化される．従って，レンダリング後の画像についてベジェ曲線の領域内をマスキングした $\mathcal{R}(\hat{I}_i)_{binary}$ を用いて， Tone Loss に適用した式 2 を使用する．

$$\mathcal{L}_{tone} = ||LPF(\mathcal{R}(I)) - LPF(\mathcal{R}(\hat{I}_i)_{binary})||_2^2 \qquad (2)$$

3.4.2 Score Distillation Sampling Loss

Score Distillation Sampling (SDS) Loss は，DreamFusion [12] において提案された，拡散モデルのヤコビアン項を省略した場合におけるサンプリング結果を用いた損失関数である．より直感的には，最小化することで条件テキストプロンプトに適合するようにパラメータ θ を最適化する損失といえる．

各反復 $t \in 1, 2, ..., T$ においてランダムにレンダリングされた画像 x が生成され，その後この画像にノイズが加えられ，ノイズスケジュールを制御する項 $\epsilon \sim \mathcal{N}(0, I)$, α_t, σ_t を伴って，$x_t = \alpha_t x + \sigma_t \epsilon$ が形成される．次に，ノイズが含まれた画像が Imagen [16] の事前学習済み UNet [15] モデルに渡され，ノイズ ϵ の予測を出力する．このとき，SDS Loss は以下の式 3 で定義される．

$$\nabla_\theta \mathcal{L}_{SDS} = E_{t,\epsilon}\left[w(t)\big(\hat{\epsilon}_\phi(x_t, t, y) - \epsilon\big)\frac{\partial x}{\partial \theta}\right] \qquad (3)$$

ここで，$\hat{\epsilon}_\phi$ は UNet に基づくノイズ除去ネットワーク，y は条件テキストプロンプト，θ は NeRF のパラメータ，$w(t)$ は α_t に依存する定数乗数である．

SDS Loss の提案された DreamFusion では 3D オブジェクト生成タスクに活用されたが，VectorFusion [8] では SDS Loss をベクターグラフィックス生成タスクに利用した．VectorFusion は，Stable Diffusion により生成されたラスター画像をベクトル化したベクター画像またはランダムに初期化されたベクター画像を初期値として，DreamFusion と同様に定義される以下の式 4 で定義される．

$$\nabla_{\hat{I}} \mathcal{L}_{LSDS} = E_{t,\epsilon}\left[w(t)\big(\hat{\epsilon}_\phi(\alpha_t z_t + \sigma_t \epsilon, y) - \epsilon\big)\frac{\partial z}{\partial z_{aug}}\frac{\partial x_{aug}}{\partial \theta}\right] (4)$$

ここで，x_{aug} は CLIPDraw [3] で示されるような透視変換及びクロップによる増強であり，z は安定拡散の事前学習済みエンコーダ $\mathcal{E}$ を適用したエンコード $z = \mathcal{E}(x_{aug})$ である．本論文でも同様に SDS Loss を用いることで VectorFusion と同様に Stable Diffusion の T2I 機能を効果的に利用することでベクターグラフィックスの生成タスクに活用する．

3.4.3 Radiation Loss

自己交差問題を解消するための Xing Loss は，単体の 3 次ベジェ曲線に対して自己交差を十全に防ぐが，セグメントが連結した実際のベジェ曲線に対してはセグメント同士の交差を防ぐことが困難であり，この点において改善の余地がある．この点に関して LIVE はパスを円形に初期化することでセグメント同士の交差を未然に防いでいるが，ターゲット画像による面積ベースの初期化が行えず最適化の過程における制御点の変量が大きい本手法では，同様の初期化を行った場合にもセグメント同士の交差が生じる可能性が高い．図 7 にパスの初期化方法によりセグメント同士が交差している例を示す．

従って，前後のセグメントとの位置関係を考慮して Xing Loss を発展させた Radiation Loss を提案する．

連結されたパス全体が交差しないために必要な最も簡易的な条件は全ての制御点がいずれかの回転方向に向かって順番に配置されることである（図 8）．すなわち，c を各セグメントの始点制御点の中央値として，パス P_n を構成するすべての制御点に対して，$\angle p_n c p_{n+1} < \angle p_n c p_{n+2}$ であ

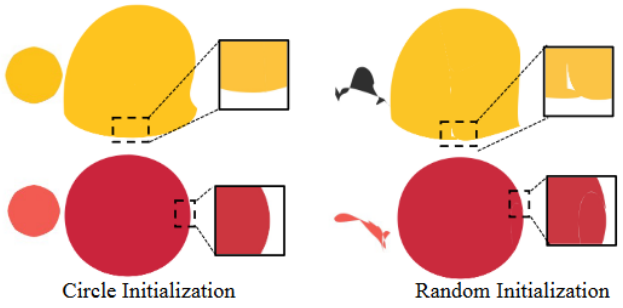

図 7 初期化方法によりパスがセグメント同士で交差している例 ([11] より引用)

ればいい．従って，Radiation Loss は以下の式 5 のように定義される．

$$\mathcal{L}_{rad} = \sum_{n} \mathrm{ReLU}(\angle p_n c p_{n+1} - \angle p_n c p_{n+2}) \tag{5}$$

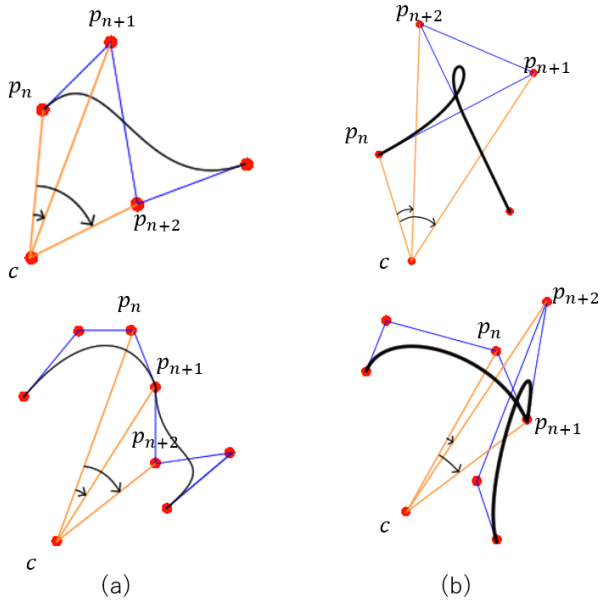

図 8 Radiation Loss の概要．(a) 回転方向に対して順番に配置した例，(b) 回転方向に逆らって配置された結果パスが交差した例．

3.4.4 Total Loss

以上の 3 つのロスを重みを伴って加算した以下の式 6 を最終の Total Loss として定義する．

$$\mathcal{L}_{total} = \lambda_{tone}\mathcal{L}_{tone} + \lambda_{rad}\mathcal{L}_{rad} + \lambda_{LSDS}\mathcal{L}_{LSDS} \tag{6}$$

なお，λ は各ロス調整するための重みである．

3.5 パスの削除

本手法ではラスタライズ後のラスター画像を用いて SDS Loss を算出するため，Loss が小さくなるようにパスの縮小，透明化が生じる．これらのパスの曖昧化はレンダリング後のラスター画像では問題にならないが，ベクター画像にとってはデータサイズおよびレンダリング処理に負の影響を与える．従って，最適化の過程でレンダリング画像に対する影響を考慮したパスの削除を行うことでより簡潔なベクター画像の生成を行う．閾値を τ として，以下の式 8 に示される条件を満たす場合に P_n を取り除く．

$$\hat{I}_{i,n} = \mathcal{R}(\{P_n, ..., p_N\}) - \mathcal{R}(\{P_{n+1}, ..., p_N\}) \tag{7}$$

$$\frac{\sum_{x,y} \mathrm{alpha}(\hat{I}_{i,n})}{w \times h} < \tau \tag{8}$$

ここで $\mathcal{R}$ はレンダラであり，$\mathrm{alpha}(\hat{I}_{i,n})$ は画像の不透明度，w と h は出力画像サイズである．式 7 は P_n がどの程度レンダリング後のラスター画像に表示されているのかを表しており，実験では $\tau = 5.0 \times 10^{-4}$ とした．図 9 にパスの削除を適用した例を示す．このように外観を損なわずにベクターグラフィックスのデータ削減が可能である．

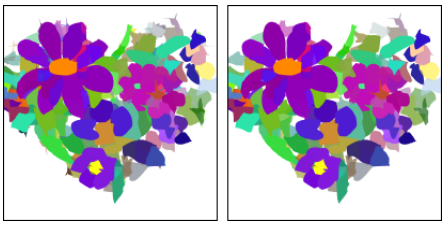

図 9 パスの削除前 (左図) とパスの削除を適用した例 (右図)．この例ではベジェ曲線が 200 本から 134 本へ減少している．

4. 実験

4.1 実験設定

入出力画像のサイズは 600×600 である．拡散モデル前の増強では $\mathcal{R}(\hat{I}_i)$ を 512×512 にクロップする．デフォルトで，ベジェ曲線の本数を 200，各セグメントの数を 6，パラメータの更新回数 i を 1000，またロスの重みをそれぞれ，$\lambda_{tone} = 200$，$\lambda_{rad} = 1$，$\lambda L_{LSDS} = 1$，さらに Tone loss 用いるローパスフィルタのカーネルサイズを 101，σ を 30 とした．なお，これらの値は経験的に設定したものである．また，$i = 800$ の時にパスの削除を行った．入力には VectorFusion [8] を参考にプロンプトエンジニアリングを用いて，“a logo of { コンセプト }. minimal flat 2d vector. lineal color. trending on artstation.” とした．

4.2 実験結果

本手法による実験結果を図 10 に示す．入力画像の形状にテキストに示される構造が生成されていることが分かる．例えば，“Mt.Fuji” の入力の場合，富士山のような構造体が出現している．最も大きい構造体としてそれぞれ炎や富士

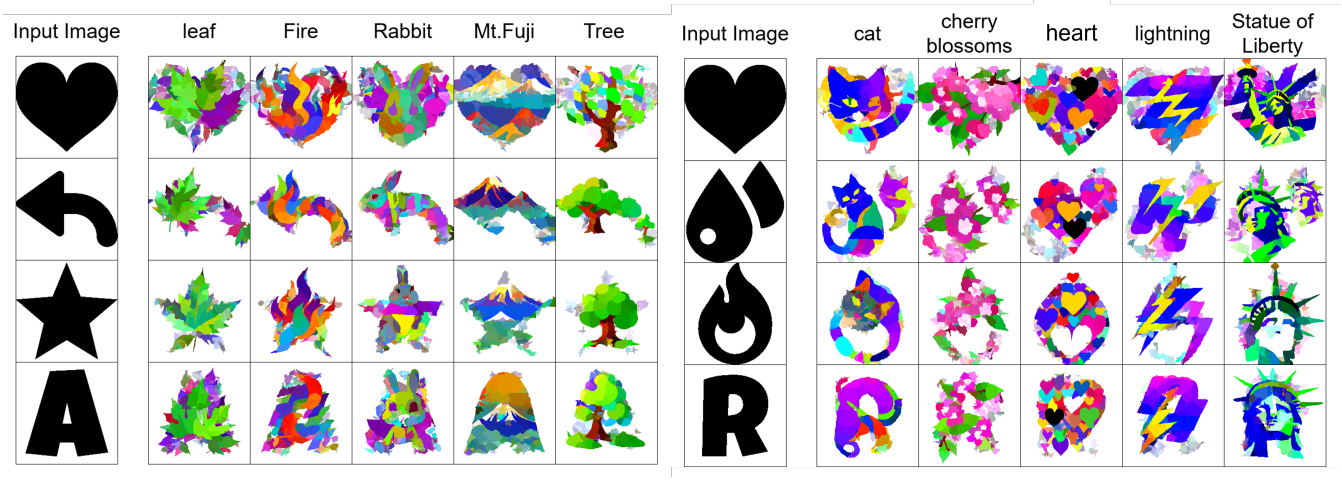

図 10 本手法による生成例

山の形状が現れている一方で入力画像領域内を満たすようにあまり意味を感じられないパスが生成されている．加えて，“rabbit” の例のように入力テキストから直感的に想像される色から離れた配色がされている．また、“lightning’ や “Statue od Liberty” の例にみられるように、青や緑などの特定の色の分布にサンプルが偏ってしまう現象が確認された。

また，比較のために CLIPDraw [3] で用いられた CLIP Loss を SDS Loss の代わりに用いた結果を図 11 に示す．“rabbit” の例や “Mt.Fuji” の例に示されるように一部にテキストに示される構造体が出現する場合があるが，SDS Loss の例に比べ全体がテクスチャ状に最適化されており，曖昧な出力となっていることが分かる．

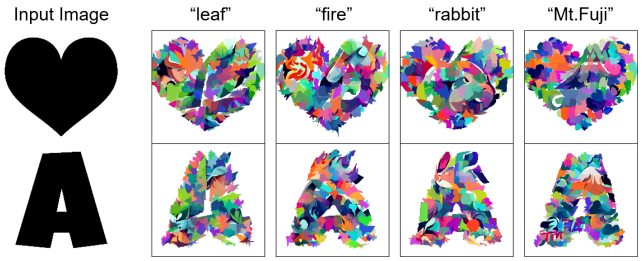

図 11 CLIP Loss を用いた場合の生成例．

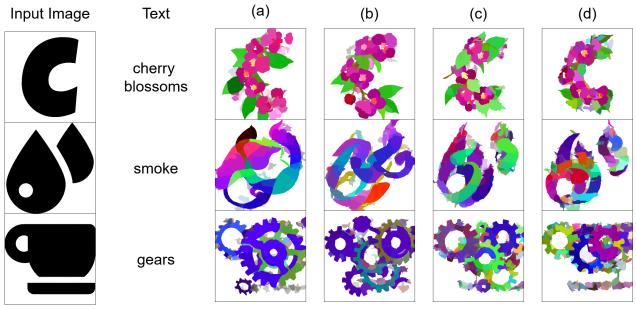

図 12 Tone Loss の重み λ_{tone} を変化させた例．(a)$\lambda_{tone} = 1.0$, (b)$\lambda_{tone} = 1.0 \times 10$, (c)$\lambda_{tone} = 1.0 \times 10^2$, (d)$\lambda_{tone} = 3.0 \times 10^2$

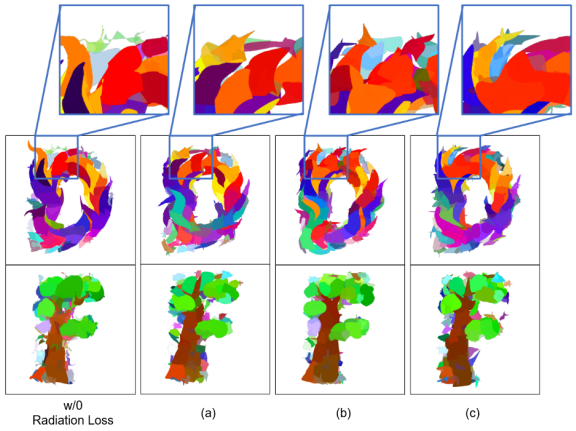

図 13 Radiation Loss の重み λ_{rad} を変化させた例．(a)$\lambda_{rad} = 1.0$, (b)$\lambda_{rad} = 1.0 \times 10$, (c)$\lambda_{rad} = 1.0 \times 10^2$

4.3 アブレーション研究

4.3.1 Tone Loss による影響

Tone Loss に異なる重みをかけた結果を図 12 に示す．重みを大きくするほど入力画像に近い形状を持つ出力が得られていることが分かる．従って，このロスを調整することによって，出力画像の形状を制御することが可能であると考えられる．

4.3.2 Radiation Loss による影響

Radiation Loss に異なる重みをかけた際の結果を図 13 に示す．Radiation Loss を取り除いた場合には自己交差による複雑なパスが生成され滑らかさが失われている一方で，Radiation Loss を追加した場合にはその重みが小さい場合でもパスが完全に凸包になっていることが分かる．ただし Radiation Loss を追加した場合にも始点制御点以外

の 2 点が大きくなることによる棘状のアーティファクトが依然として生成されている．

また，Xing Loss と Radiation Loss の比較を図 14 に示す．Xing Loss の場合に比べて Radiation Loss ではパスの自己交差が低減されており，各パスがより大きい構造を形成していることが分かる．

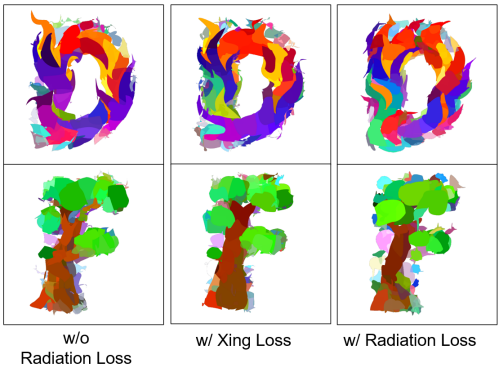

図 14 Xing Loss と Radiation Loss の比較

4.4 その他のハイパーパラメータの影響

テキストプロンプトの操作による出力結果への影響を調べるために形容詞に色を追加した場合やプロンプトのプレフィックス，サフィックスを変更して実験を行った．図 15 にテキストプロンプトを “{ 色 } { コンセプト }” として生成を行った結果を示す．有彩色の入力による出力例では色を反映している一方で，“black” や “white” などの無彩色の場合には色が無視されるか一部にのみ適用されていることが分かる．

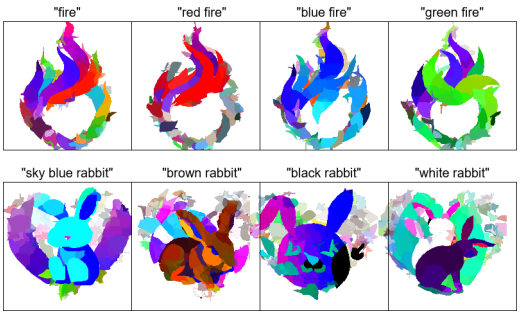

図 15 形容詞に色を追加した場合の出力例

図 16 は，パラメータの更新回数を変更した場合の出力例である．$i = 1000$ で大部分の形状が形成されており，$i = 1500$ にはほとんど完全に安定した形状が出力うされていることが分かる．従って，$1000 < i < 1500$ の間で適切な値を選択することが望ましい．

また，パスの本数 N を変更した場合の生成例を図 17 に示す．パスの削除によって，過度なパスの本数を指定した場合に不要なパスが削除されていることが分かる．

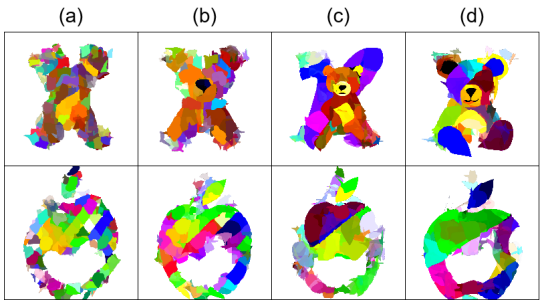

図 16 パラメータの更新回数 i を変更した場合の出力例．(a)$i = 500$，(b)$i = 1000$，(c)$i = 1500$，(d)$i = 2000$，上段の入力テキストが “bear”，下段が “apple” である．

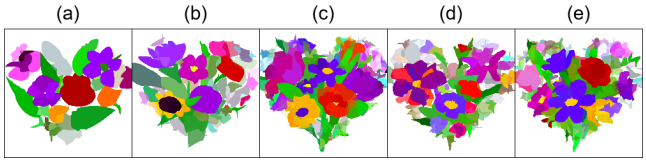

図 17 パスの本数 N を変化させた場合の出力例．パスの削除を適用することによって以下のようにパスの本数が変化する．(a)32 本から 32 本，(b)64 本から 52 本，(c)128 本から 116 本，(d)256 本から 141 本，(e)512 本から 166 本．

5. 考察

5.1 出力結果について

出力結果のサンプル（図 10）を観察すると，一部にテキストに示されるような構造体が出現する一方で，その他の部分には形状を満たすようにあまり意味のないパスが出現している．これは SDS Loss で得られるサンプル結果と形状画像の乖離を Tone Loss によって制限しているために起こる減少だと考えられる．また，“fire” や “rabbit” に示されるようにテキストから想像される色から乖離しているがこれは SDS Loss によって、ランダム生成された状態を引き継いだパスが入力画像の領域を満たすために残留している可能性があることや，プロンプトのプレフィックスやサフィックスに起因する構造体が形状の制限によって破壊されている可能性が考えられる．これらの形状の破壊に関して Tone Loss による形状の制限では SDS Loss によるサンプリング結果の形状を変化させることが不可能であることから，入力画像の形状にクロップしているのと同じような効果になっているのではないか考えられる．従って，入力画像を拡散モデルの条件付けに用いるなどの方法によって，サンプリング結果そのものがある程度入力の形状を維持できるにする拡張が必要である．

5.2 Radiation Loss の効果

図 13 に示されるように Radiation Loss は自己交差問題を抑え，パスの形状の安定化に一定の効果が示された．その一方で完全に滑らかなパスの生成には未だ不十分であり，特定のセグメントが棘状に伸びたアーティファクトが形成されている．これは，Radiation Loss の制約の中で不要なセグメントによる描画への影響をおさえるために描画面積を少なくするように最適化されたためだと考えられる．また，Radiation Loss は各セグメントの始点制御点に関する情報が含まれているため，始点制御点の位置に不必要な制限が設けられ，不可能なパスの形状が生じる問題がある．本手法ではレンダリング後のラスター画像によりコンテンツを示す SDS Loss を使用しているため，複数のベジェ曲線による複合の効果で不可能な形状へ対処されているが，これは適切に整理されたベクターパスという目的にはそぐわない．従って Radiation Loss における中心点 c に相当する適切な基準位置の設定が必要である．

6. まとめ

本論文では，微分可能レンダラー [9] の機能と SDS Loss[12] 及び Tone Loss[6] を組み合わせることで形状を維持したベクターグラフィックスの生成手法を提案した．さらに，Xing Loss[11] を拡張した Radiation Loss を導入することで自己交差問題に対処した．実験結果より本手法では入力画像の形状を維持したままテキストの内容をベクターグラフィックスに反映することが可能である一方，入力の条件に出力結果が左右されるため依然改良の余地が残る．

今後の展望として, 形状を直接制御するのではなく Stable Diffusion の Image-to-Image による条件付け等を用いてサンプリング結果そのものを入力画像に近づける方法が考えられる．また，Radiation Loss における中心座標 c の代わりに中心軸変換 [2] を用いる拡張を行うことで始点制御点に対する不要な制限を緩和する．

参考文献

[1] Atarsaikhan, G., Iwana, B. and Uchida, S.: Contained Neural Style Transfer for Decorated Logo Generation, *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*, pp. 317–322 (2018).
[2] Blum, H.(ed.): *A Transformation for Extracting New Descriptors of Shape*, pp. 362–380, MIT Press (1967).
[3] Frans, K., Soros, L. and Witkowski, O.: CLIPDraw: Exploring Text-to-Drawing Synthesis through Language-Image Encoders, *Advances in Neural Information Processing Systems*, Vol. 35, pp. 5207–5218 (2022).
[4] Gihyun, K. and Jong Chul, Y.: Clipstyler: Image style transfer with a single text condition, *Procs. of IEEE Computer Vision and Pattern Recognition*, pp. 18062–18071 (2022).
[5] Hertzmann, A.: Painterly rendering with curved brush strokes of multiple sizes, *Procs. of the 25th annual conference on Computer graphics and interactive techniques - SIGGRAPH '98*, pp. 453–460 (1998).
[6] Iluz, S., Vinker, Y., Hertz, A., Berio, D., Cohen-Or, D. and Shamir, A.: Word-As-Image for Semantic Typography, *ACM Transactions on Graphics*, Vol. 42, No. 4, pp. 1–11 (2023).
[7] Izumi, K. and Yanai, K.: Zero-Shot Font Style Transfer with a Differentiable Renderer, *Procs. of the 4th ACM International Conference on Multimedia in Asia*, pp. 1–5 (2022).
[8] Jain, A., Xie, A. and Abbeel, P.: VectorFusion: Text-to-SVG by Abstracting Pixel-Based Diffusion Models, *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1911–1920 (2023).
[9] Li, T., Lukáč, M., M, G. and Ragan-Kelley, J.: Differentiable Vector Graphics Rasterization for Editing and Learning, *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, Vol. 39, No. 6, pp. 193:1–193:15 (2020).
[10] Li, T.-M., Aittala, M., Durand, F. and Lehtinen, J.: Differentiable Monte Carlo Ray Tracing through Edge Sampling, *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, Vol. 37, No. 6, pp. 222:1–222:11 (2018).
[11] Ma, X., Zhou, Y., Xu, X., Sun, B., Filev, V., Orlov, N., Fu, Y. and Shi, H.: Towards layer-wise image vectorization, *Procs. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16314–16323 (2022).
[12] Poole, B., Jain, A., Barron, J. T. and Mildenhall, B.: DreamFusion: Text-to-3D using 2D Diffusion, *The Eleventh International Conference on Learning Representations* (2022).
[13] Radford, A., Kim, J., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J. et al.: Learning transferable visual models from natural language supervision, *International Conference on Machine Learning*, pp. 8748–8763 (2021).
[14] Rombach, R., Blattmann, A., Lorenz, D., Esser, P. and Ommer, B.: High-Resolution Image Synthesis with Latent Diffusion Models, *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10674–10685 (2022).
[15] Ronneberger, O., Fischer, P. and Brox, T.: U-net: Convolutional networks for biomedical image segmentation, *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference*, pp. 234–241 (2015).
[16] Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E., Ghasemipour, S. K. S., Gontijo-Lopes, R., Ayan, B. K., Salimans, T. et al.: Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding, *Advances in Neural Information Processing Systems* (2022).
[17] Schaldenbrand, P., Liu, Z. and Oh, J.: StyleCLIPDraw: Coupling Content and Style in Text-to-Drawing Translation, *Procs. of the Thirty-First International Joint Conference on Artificial Intelligence*, pp. 4966–4972 (2022).
[18] Selinger, P.: Potrace : a polygon-based tracing algorithm (2003). http://potrace.sourceforge.net/potrace.pdf.