第4章

提案手法

本章では，まず4.1節において提案手法の説明に用いる用語の定義を示す．次に，4.2節で提案手法がデザイン文字列を自動作成する処理の概要を説明する．その後，4.3節で提案手法が自動作成の対象とするデザイン文字列や，提案手法の入力として用いる画像が満たすべき条件を明示する．本章の最後に，4.4節で4.2節で述べた処理の実装について述べる．

4.1 用語の定義

本手法の説明で用いる用語の定義を次の表 4.1に示す．

表4.1 用語の定義

用語	定義
コンテンツ	出力されるデザイン文字列で使用したい文字列
歪みのスタイル	出力されるデザイン文字列に付与したい形状の歪み
エフェクトのスタイル	出力されるデザイン文字列に付与したい色やエフェクト
デザイン文字のグリフ	デザイン文字に対し，色やエフェクトを無視した文字形状

また，以下では「バウンディングボックス」を「B.B.」と略記する．さらに，画像の座標系は左上隅を原点，右向きをx軸の正の向き，下向きをy軸の正の向きとする．

4.2　デザイン文字列の自動作成処理の概要

本手法では，次の図 4.1のようにしてデザイン文字列を自動作成する．

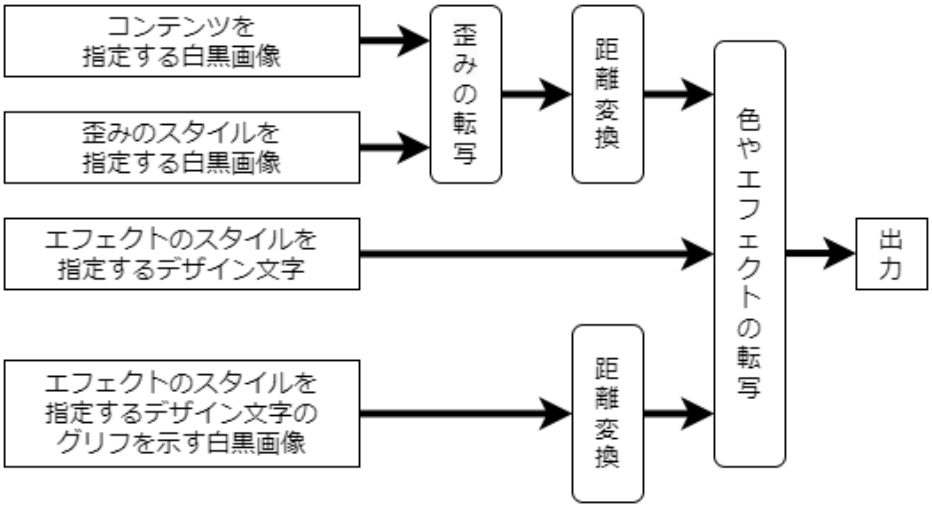

図4.1　デザイン文字列の自動作成処理の流れ

図 4.1に示した処理の流れを順に説明する．

まず，歪みのスタイルを指定する文字列画像からコンテンツを指定する文字列画像へと文字列の形状の歪みを転写する処理を行う．この処理は歪みのスタイルを指定する文字列画像の形状の歪みを分析した後，その歪みを再現するように射影変換を用いてコンテンツを指定する文字列画像を歪ませるというような手順で実行する．（後の4.4.1節で詳述する．）

次に，先行研究[1]で提案されたモデルを用いてエフェクトのスタイルを転写するために，歪みの転写で得た文字列画像と，エフェクトのスタイルを指定するデザイン文字のグリフを示す画像に対して距離変換と呼ばれる処理を行う．

最後に，得た二枚の距離変換画像とエフェクトのスタイルを指定するデザイン文字から，Style Transferを用いて最終的な出力となるデザイン文字列を作成する．以上が，本手法におけるデザイン文字列の自動作成処理の大まかな流れである．

4.3　対象のデザイン文字列

本手法では，デザイン文字列のうち文字の配置が横一行のもののみを対象とする．つまり，**??**で述べたようなデザイン文字列に特有のスタイルのうち，文字列全体の形状の歪みのみを本手法の対象とし，文字の配置や文字の間隔などは対象外とする．

但し，形状の歪みによって文字配置が横一直線でなくなったものも本手法の対象とする．例えば，次の図 4.2の文字配置は傾いているが，こうした文字列も本手法の対象とする．

図4.2　形状の歪みにより，文字配置が傾いた文字列の例

さらに，歪みのスタイルの転写元となる画像は次の二つの条件を満たす必要がある．

条件1: 画像中の適切な位置で，適切な回数だけ縦に画像を区切ることで左右に画像を分割すれば，分割された各領域内の形状の歪みは単純な射影変換で再現できる．

条件2: 条件1で述べた適切な画像の分割回数が文字数に比べて十分少ない．

歪みのスタイルの転写元の画像が条件1を満たすとき，分割された各領域内では，形状の歪みは単純な射影変換で再現できることから，画像中に含まれる各文字のB.B.の中心のx座標に対し， B.B.の中心のy座標とB.B.の大きさがほぼ線形とみなせるといえる．

例えば，次の図 4.3aは 図 4.3bのように補助線を引くと，「P」の付近で左半分の領域と右半分の領域の2つの領域に分けることができる．このとき，左半分の領域の文字列の形状の歪みと右半分の領域のの文字列の形状の歪みは，どちらも単純な射影変換で再現できるため，もとの図 4.3aの画像は条件1を満たすといえる．

(a) もともとの文字列画像

(b) 補助線を引いた文字列画像

(c) バウンディングボックスつきの文字列画像

図4.3　条件1を満たす文字列画像の例

一方，図 4.3bと図 4.3cを見比べると，左半分の領域では右に行くほどB.B.の大きさがほぼ線形に小さくなっており，右半分の領域では右に行くほどB.B.の大きさがほぼ線形に大きくなっているとわかる．また，B.B.の中心のy座標は図 4.3cを見る限りほぼ一定である．つまり，この図 4.3aの画像においては，画像に含まれる文字のB.B.の中心のx座標に対し，B.B.の中心のy座標とB.B.の大きさが区分的にほぼ線形であるといえる．

この例からは，歪みのスタイルの転写元の画像が条件1を満たすとき，画像に含まれる文字のB.B.の中心のx座標に対し， B.B.の中心のy座標とB.B.の大きさが区分的にほぼ線形となり，区間の境界と形状の歪み方が切り替わる位置がほぼ一致することが予想できる．

この予想が正しいと仮定すると，歪みのスタイルの転写元の画像として図 4.3aのような画像が与えられたとき， 図 4.3cのように各文字のB.B.を計算した後に，得たB.B.の中心のx座標に対し，B.B.の中心のy座標とB.B.の大きさを区分線形回帰することで，区間の境界位置として形状の歪みの切り替わる位置を推定し， 図 4.3bのような形状の歪みの情報を計算することができる．これが歪みのスタイルの転写元の画像が条件1を満たすべき理由である．

また，歪みのスタイルの転写元の画像が条件2を満たすとき，分割された各領域内には十分多くの文字が含まれることになる．これは上述したように区分線形回帰を用いて形状の歪みの切り替わる位置を推定する際に，回帰の精度を保証するために必要な条件であるため，歪みのスタイルの転写元の画像は条件2も満たすべきである．

4.4 実装

4.4.1 歪みのスタイルの転写

歪みのスタイルの転写処理は4.2節で簡単に述べたように，歪みのスタイルの転写元の画像を分析して目的とする歪みのスタイルの情報を得る処理と，得た情報を元に歪みのスタイルの転写先の画像を射影変換を用いて歪ませる処理の，二つに大きく分けられる．

このうち，歪みのスタイルの情報を得る処理は， 4.3節で歪みのスタイルの転写元の画像が 2つの条件を満たすべき理由として述べたように区分線形回帰を用いて実行する．

但し，4.3節では分割された領域内の形状の歪みが同じになるように領域を分割するとき，分割後の領域数として最適な値の決定方法については考慮していなかった．そのため，実際に歪みのスタイルの転写元の画像のみから歪みのスタイルの情報を得る際には，分割後の領域数として最適な値を自動的に決定できるようにアルゴリズムを修正しなければならない．

そして，修正したアルゴリズムの疑似コードが次のAlgorithm 1である．このアルゴリズムでは，分割後の領域数を1から初めて徐々に増やしながら繰り返し区分線形回帰を実行していく．そして，回帰の二乗平均平方根誤差が閾値以下となるか，ひとつ前のループの回帰の二乗平均平方根誤差との差分が閾値以上であった場合，そのループにおける区分線形回帰の結果を採用して計算を終了する．逆に，ひとつ前のループに比べて二乗平均平方根誤差が悪化した場合は，ひとつ前のループの区分線形回帰の結果を採用して計算を終了する．また，分割後の領域数が画像の文字数の半分を上回った場合は計算を強制終了し，最後のループにおける区分線形回帰の結果を採用する．なお，区分線形回帰の実装にはpythonのライブラリであるpiecewise linear fittingを用いた．以上のようにして得た歪みのスタイルの情報を元に，スタイルの転写先の画像を射影変換を用いて歪ませることで歪みのスタイルを転写する．なお，射影変換の実装にはpythonのライブラリであるOpenCV を用いた．

Algorithm 1 歪みのスタイルの情報の推定

Input:

$warped\_image$:歪みのスタイルを指定する白黒画像
$\Delta\varepsilon$:区分線形回帰の二乗平均平方根誤差の差分の閾値
ε: 区分線形回帰の二乗平均平方根誤差の閾値

Output:

fit_result:区分線形回帰の結果

function TRANSFER_WARP_STYLE($target\_image, warped\_image, \Delta\varepsilon, \varepsilon$)
    $bounding\_boxes$ = GET_BOUNDING_BOXES($warped\_image$)
    num_boxes = GET_LENGTH($bounding\_boxes$)
    $num\_segments = 1$
    while $num\_segments \leq num\_boxes/2$ **do**
        fit_result = PIECEWISE_LINEAR_FIT($bounding\_boxes, num\_segments$)
        $error$ = GET_FITTING_ERROR(fit_result)
        if $num\_segments \neq 1$ **then**
            if $prev\_error - error < 0$ **then**
                $fit\_result = prev\_fit\_result$
                break
            else if $prev\_error - error > \Delta\varepsilon$ **or** $error \leq \varepsilon$ **then**
                break
            end if
        end if
        $num\_segments = num\_segments + 1$
        $prev\_fit\_result = fit\_result$
        $prev\_error = error$
    end while
    return fit_result
end function

4.4.2 距離変換およびエフェクトのスタイルの転写

エフェクトのスタイルの転写処理は4.2節で簡単に述べたが，先行研究[1]で提案されたモデルを用いて実行する.

参考文献

[1] W. Wang, J. Liu, S. Yang, and Z. Guo, “Typography with decor: Intelligent text style transfer,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019.