第4章

提案手法

本章では，まず4.1節で提案手法の処理の流れを説明する．次に，4.2節で対象とするデザイン文字列を明示する．本章の最後に，4.3節で4.1節で述べた処理の実装について述べる．

4.1　デザイン文字列を自動作成する処理の流れ

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．

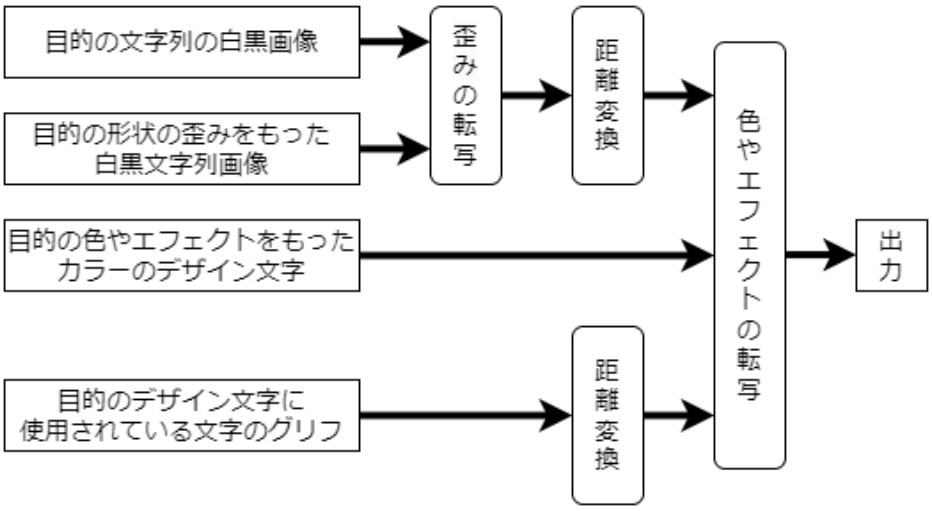

図4.1　デザイン文字列の自動作成処理の流れ

図 4.1に示した処理の流れを順に説明する.

本手法を用いてデザイン文字列を作成するために必要な入力データは次の４つである.

入力1: 出力されるデザイン文字列において使用したい文字列の白黒画像

入力2: 出力されるデザイン文字列全体の形状の歪みを指定するための白黒文字列画像

入力3: 出力されるデザイン文字列の色やエフェクトを指定するためのカラーのデザイン文字

入力4: 入力3のカラーのデザイン文字に使用されている文字のグリフ

これらのうち，まずは入力2のもつ形状の歪みを分析して入力1に形状の歪みを転写することで，目的の形状の歪みを持った目的の文字列の白黒画像を作成する．次に，形状の歪みの転写を用いて作成した白黒文字列画像と，入力4に対してそれぞれ距離変換を施す．最後に，得た二枚の距離変換画像と入力3から最終的に出力す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流れである.

4.2　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　デザイン文字列の自動作成処理の流れ

さらに，本手法での歪みの転写の実装上，入力2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，縦線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．

二つの条件のうち，条件1は目的とする形状の歪みが射影変換を用いて再現可能であるための必要条件を表している．例えば，次の図 4.3の文字列はいくら画像を縦線で左右に分割したとしても，分割された各領域内の形状の歪みは単純な射影変換では再現できないため，本手法の対象ではない．

図4.3　本手法の対象ではない形状の歪みをもった文字列画像の例

一方，次の図 4.4aの文字列は画像の中央を通る縦線（図 4.4bの赤線）で左右に画像を分割すると，分割された各領域内の形状の歪みが単純な射影変換で再現できるため，本手法の対象となる．

(a) 補助線なし　　**(b)** 補助線あり

図4.4　本手法の対象の形状の歪みをもった文字列画像の例

また，二つの条件のうち条件2は，本手法において目的とする形状の歪み方がうまく検出されるために必要な条件を表している．（後の4.3.1節で詳述する．）

4.3 実装

4.3.1 文字列の歪みの転写

4.1節で述べた処理のうち，文字列の歪みの転写は次のように行っている．

バウンディングボックスの検出

歪みの数を決定する（後述）

区分線形回帰

歪みの数の決定方法

制御点の位置を計算

アルゴリズム **1** 文字列の歪みの転写

```
1: function MAX_IN_ARRAY(array)
2:     max ← 0
3:     for all element ← array do
4:         if element > max then
5:             max ← element
6:         end if
7:     end for
8:     return max
9: end function
```

4.3.2 距離変換および色やエフェクトの転写

先行研究**??**で提案されたモデルを用いる．