重み付き有限状態トランスデューサを用いた情景画像からの日本語語彙検出

山添　隆文† 　栄藤　稔† 　吉村　健† 　辻野　孝輔†

† 株式会社 NTT ドコモ　〒 239–8536 神奈川県横須賀市光の丘 3–6

E-mail: †yamazoet@nttdocomo.com, ††{etoh,yoshimura.takeshi,tsujino}@nttdocomo.co.jp

あらまし　本稿では，重みつき有限状態トランスデューサ（WFST）を用い，ノイズ領域が文字列として誤検出されることが課題となる情景画像からの日本語語彙検出手法を提案する．提案手法では，情景画像から網羅的な文字候補を抽出し，それらすべての組み合わせを表現した WFST と，大規模な語彙データセットから辞書を構成する WFST とを合成することより，最終的な語彙検出結果を得る．これにより，辞書に含まれない文字候補の組み合わせとなるノイズ領域を除去することが可能となる．また，情景画像からの文字候補の抽出において，画像における位置情報を保持したまま WFST を生成するため，同一の画像領域から最適な語彙を検出することができる．日本語を含む情景画像のデータセットにより検証を行い，ノイズ領域を除去しつつ日本語の語彙を検出できることを確認した．

キーワード　情景画像，文字認識，文字領域検出，WFST

1. はじめに

近年，情景画像中からの文字認識が発展しつつあり，Evernote [1] のように検索機能等を提供するサービスの利用が拡大している．

上記のようなサービスは，モバイル機器で撮影された画像を活用するクラウドベースの基盤として今後さらなる発展が期待されており，従来のモバイル機器では利用できなかった大容量の辞書やデータベース，処理速度・大量のメモリを必要とする高負荷な処理も可能となっている．しかし，情景画像からの文字認識では，図 1 のように，文字の方向，サイズ，色，フォント等が自明ではなく，木の葉のように検出された画像領域が文字と似た形状となってしまう場合がある．

数年前に ICDAR で行われた情景からの文字検出のコンテストでは [2] [3]，テストセットに対する評価が行われた．これらのコンテストでは，Adaboost や SVM といった統計的手法や機械学習を基本とした，高度な前処理や領域検出が行われている [4] [5] [6]．これらの前処理を適用した後，図 2(a) のように OCR 処理を行い，後処理としてエラー訂正が行われているのが一般的である．

それに対し，本稿では，大規模かつ複雑な状態遷移の合成を高速に行うことができる重み付有限状態トランスデューサ（WFST）[7] の特性を利用した情景画像からの日本語の語彙検出手法を提案する (図 2(b))．

提案手法では，情景画像から文字候補を抽出した文字認識結果の WFST と，語彙セットを構成する辞書の WFST の二つの合成演算により語彙検出を実現する．文字認識結果の WFST については，誤検出を許容して極力取りこぼしがなくなるような大量の文字列候補・文字候補群を取得し，WFST で表現する．一方，辞書については，出現頻度等の重みのついた語彙群から構成される WFST として表現する．この二つの WFST を合成することで，情景画像中からノイズとなる文字認識結果を除去しつつ，目的の語彙の検出を行うことが可能となる．

本手法により，情景画像からの文字検索用テーブルの作成やタグ情報生成，さらにはデータマイニングのためのジャンルを限定した語彙検出等，様々なアプリケーションへの応用が期待される．

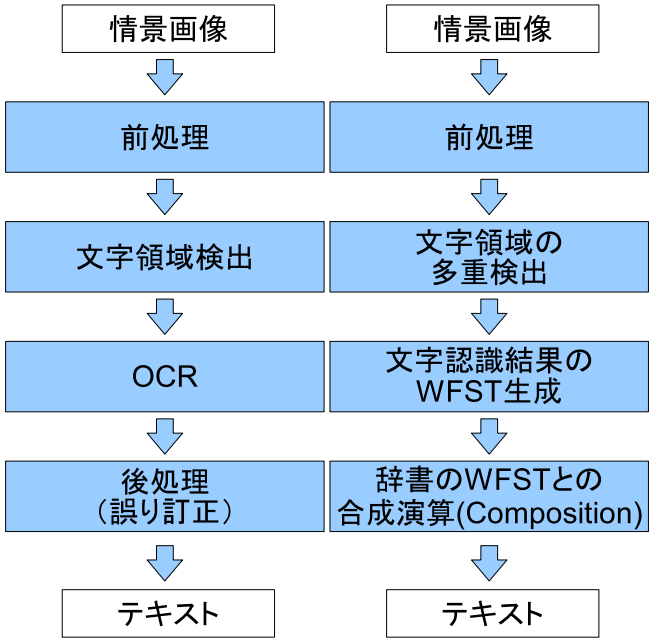
WFST を用いた文字認識の従来研究としては，単語単位に文字列が分割されている状態から文字認識エンジンにより 3 つの候補を取得し，語彙の辞書との WFST 合成演算によるベストパス算出により推定を行う手法を提案されている [13]．この手法では，演算結果の重みが閾値を下回る場合は認識候補の WFST のベストパスを語彙とし，文字形状による候補の修正を行っている．また，切り出された単語に対し，オーバーセグメンテーションによる文字切り出し・文字認識を行い，WFST で語彙を検出する手法が提案されている [14]．この手法では，文字認識結果の尤度，切り出し処理における切り出し位置である確率，切り出し幅から遷移の重みを計算している．また，オープンソースの OCR エンジン Ocropus でも WFST による言語処理がフレームワークとして採用されている [15]．日本語文字認識についても，ドキュメント OCR の結果を WFST により修正する手法も提案されている [16]．しかし，情景画像からの日本語文字認識を考えた場合，日本語は漢字，かたかな，ひらがな等，構成する文字がアルファベットと比べて非常に多い上，文字のデザインや種類が様々であるため，認識結果の文字形状の特徴による修正が OCR に比べて困難である．

提案手法では，日本語のような分かち書きでない言語（単語と単語の間にスペースのようなセパレータがない

(a) ビルの標識
（左折禁止・止まれ）

(b) 森林の看板
（名水百選・龍ケ窪・飲用水）

図 1　情景画像の例



(a) 従来手法

(b) 提案手法

図 2　提案手法の処理の流れ

記述）と英語のように分かち書き言語が混在した情景画像で，文字認識ではノイズとなる領域が誤検出されていることを前提としている．

そのため，提案手法による語彙の検出は従来手法で行われているような最上位パスの遷移を取得するのではなく，WFST として空間的な配置を保持する構造とし，画像全体から正解語彙として可能性が高い順に語彙の配置が得ることで，文字列領域としての誤検出や文字列内にノイズとなる領域が多く現れる場合でも語彙の検出を行うことができる．

2.　文字領域検出・文字認識処理

文字領域検出および文字認識については本稿における主題ではないが，WFST の大規模かつ複雑な状態遷移に対する演算を高速に行える特性を用いるために有効な，多重仮説の生成に重要な役割を持つ．そのため，文字領域は誤検出を減らすこと (Precision) よりも，検出に成功すること (Recall) を優先する．

2.1　文字候補領域の検出

文字の候補となる領域の検出については，NLNiblack 法による二値化と connected-component（CC）による文字領域判定手法を利用する [8]．上記手法では，CC として NLNiblack 法による二値画像から得られる形状の特徴を Adaboost による判定で絞り込んでいるが，提案手法では Adaboost 等によるパラメータ調整を行っておらず，より多くの仮説が生成することで，より網羅的に文字の候補となる領域の検出を目指す．(1)〜(4) に文字候補領域検出の手順を示す．

（1）　バイリテラルフィルタ，膨張収縮処理によりノイズ除去

（2）　NLNiblack 法により白い背景と黒い背景のための二値化を行う．なお，二値化については，画像サイズを変える等の複数のパターンによる処理結果を用いる

（3）　二値化した画像のほか，複数領域で構成される文字を検出するために 3 x 3 膨張処理，5 x 5 膨張処理を行った画像，ノイズにより背景と接続された文字を検出するために 3 x 3 収縮処理を行った画像，のそれぞれに対してラベリング処理を行う

（4）　各候補領域毎に文字らしい形状として，

- 候補領域の円形度
- 候補領域矩形における候補領域面積とそれ以外の領域の面積の比率
- 候補領域を構成する領域数（膨張処理の場合）
- 候補領域中の穴の数
- アスペクト比，面積，矩形サイズ等

から文字領域判定を行う

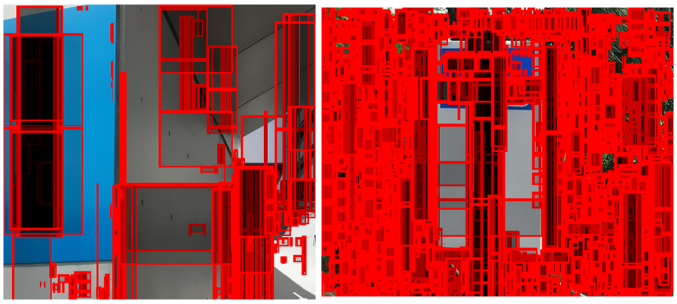
2.2　候補領域からの文字列検出

候補領域からの文字列の検出については，検出された CC の領域に対するモルフォロジー演算による領域検出が提案されている [9]．また，空間的な文字の配置からせん断変形のパラメータを推定する手法が提案されている [10]．これらの手法を組み合わせ，近接する局所的な文字候補の配置から情景画像中からせん断変形を補正した文字列領域の矩形を取得する．(1)〜(4) に文字列検出の手順を示す．

（1）　任意の 3 点の文字領域について，文字列の傾きが±３０度（縦書きは＋ 90 度）以内，連続する 3 点のなす角度の変化が±１０度以内，領域間の距離が領域サイズの 2/3 以上，2 倍以下，領域サイズが 1/2 以上，2 倍以下の条件を満たすものを文字列領域として求める

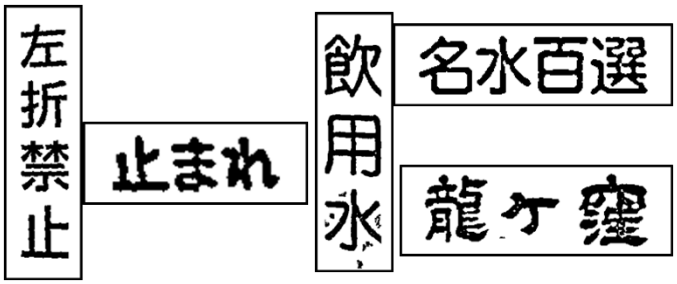
（2）　上記文字列領域で同一の文字領域を持つものを連結し，ラベリングする

（3）　文字列領域の傾きの中央値から±５度を超えるものを除去し，残った領域を再ラベリングし，文字列領



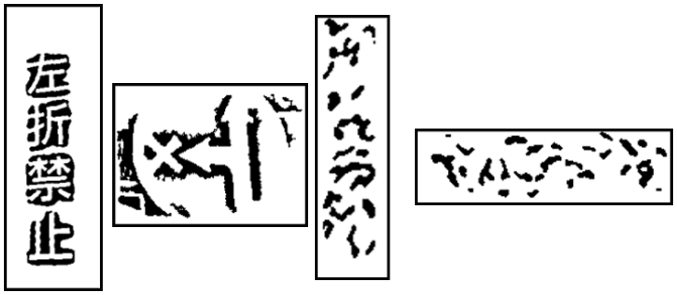
(a) ビルの標識　　(b) 森林の看板

図 3　検出された文字列領域 (赤枠)



(a) ビルの標識　　(b) 森林の看板

図 4　文字列検出成功例



(a) ビルの標識　　(b) 森林の看板

図 5　文字列検出失敗例

域の矩形と傾きを求める

（4） ノイズと矩形外周につながる領域を除去し，文字列の傾きを元に垂直方向のせん断変形を補正し，文字間の隙間数と面積が最大となるせん断変形の角度を探索することで，文字列のせん断・回転変形を補正する

図 3 に情景画像から検出された文字列の領域を示す．多重仮説による文字候補領域から，縦方向・横方向の文字列領域を取得しているため，図 4 のように方向や大きさ，文字色の違う領域を，文字列領域として検出できている．その反面，高度な文字領域判定処理を行っていないため，図 5 のような二値化に失敗している文字列領域，図形，木の葉等が文字列として大量に誤検出されている．

2.3　文字列からの文字の切り出し

文字の切り出しは，サーベイ文献 [11] に示されるように様々な手法が提案されているが，提案手法では，各文字列矩形に対しせん断・回転補正を行ったうえで，垂直方向の幅と画素のヒストグラムから微分値の閾値を超えた位置を切り出し位置として利用し，空間的に重複する文字の切り出し位置の決定を行う（オーバーセグメンテーション）．

(1)〜(6) に横書きの場合の文字切り出し処理の手順を示す．縦書きの場合は９０度回転して同様の処理を行う．

（1） 垂直方向 h のヒストグラム ($projection_{pixel}$) を作成

（2） 垂直方向 h の画素を走査し，空白でない画素の最大値と最小値から垂直方向の幅（$projection_{profile}$）を求める

（3） 式 (1) で切り出し位置を求めるためのヒストグラム h_{th} を計算する

$$h_{th} = \frac{projection_{profile}}{h} + 2\left(\frac{projection_{pixel}}{h}\right)^2 \quad (1)$$

（4）（3）のヒストグラムで１を超える値を閾値として最小値となる位置と空白位置となる位置を切り出し位置とする

（5） 文字列検出で求めた文字領域サイズの中央値を元に切り出し位置を決定し，（4）の切り出し位置に加える

（6） 両端が空白となる切り出しを除外し，平均切り出し幅の 0.125 倍以上，1.75 倍以下のサイズで領域が重複する形で切り出し，多重仮説のための状態遷移を生成する

2.4　文字認識

文字認識は文字列画像から切り出された文字の候補領域に対して一文字単位で行う．提案手法を適用するにあたっては特定の文字認識エンジンに依存しておらず，本稿では，文字認識エンジンとしてオープンソースで公開されている NHOCR [12] と商用の日本語 OCR エンジンを利用する．文字認識は，切り出された文字からノイズ除去をせずに行うほか，文字サイズを 80 x 80 に正規化し，メディアンフィルターをかけたものに対しても行う．NHOCR の出力としては，20 候補（候補数固定），商用 OCR エンジンでは最大 8 候補（候補数は変動する）とし，フィルタリングあり・なしの２パターンずつ，最大５６候補の認識結果を一文字毎に取得する．

また，オーバーセグメンテーションによる多重仮説生成を行っているため，複数の文字を一文字として認識する場合があるので，NHOCR の文字認識は一文字の認識結果として複数文字による結果が表れることを許容する．また，商用 OCR エンジンについては文字列認識機能があるため，一文字単位の認識だけでなく，文字列に対する認識結果も候補群として利用する．

図 4 の文字認識結果について単純な認識結果第一候補

を見ると，(a)「左折禁止」・「止まか」，(b)「龍ゲ褒」・「名水百選」・「飲用氷」という結果になり，フォントや二値化のノイズ等により誤推定があらわれているが，下位候補には正解文字が含まれている．

3. WFST 処理

画像中から網羅的に文字の領域とその文字認識結果の候補群を検出することについて述べたが、辞書にある単語として結果を取り出す場合、大規模な語彙の辞書の中から正解を検出するために膨大な組み合わせの探索を行う必要がある．そこで，高い拡張性を持った汎用的な状態遷移の表現が行え，大規模かつ複雑な状態遷移の合成を高速に行うことができる，重み付有限状態トランスデューサ（WFST）を利用する．

WFST は，記号列変換と重みの集合を表現したもので，入力記号列を出力記号列と重みの集合に変換する操作について式 (2) により表すことができる．

$$
\begin{aligned}
&T = (\Sigma, \Delta, Q, K, E, I, F)\,,\\
&\Sigma\text{: 入力記号の有限集合,}\\
&\Delta\text{: 出力記号の有限集合,}\\
&Q\text{：状態の有限集合,}\\
&K\text{：重みの半環（}Semiring\text{）,}\\
&E : Q \times (\Sigma \cup \{\ \varepsilon\ \}) \times (\Delta \cup \{\ \varepsilon\ \}) \times K \times Q :\\
&\qquad \text{遷移の有限集合,}\\
&\lambda\text{: } I \to K\text{：初期状態重み関数,}\\
&\rho\text{: } F \to K\text{：最終状態重み関数,}\\
&I \subseteqq Q\text{：初期状態の集合,}\\
&F \subseteqq Q\text{：最終状態の集合,}\\
&\varepsilon\text{：入力もしくは出力における空の遷移を表す特殊記号.}
\end{aligned}
\tag{2}
$$

また、二つの WFST の合成演算を行うことで，両者の WFST に共通する状態遷移とその重みを取得することができる．WFST の合成演算は式 (3) のように表され，この演算により 2 つ以上の WFST の入出力の変換を合成する

$$
\begin{aligned}
&T = T_{OCR} \circ T_{LEX}, where\\
&T_{OCR}{=}(\Sigma_{OCR},\Delta_{OCR},Q_{OCR},K_{OCR},E_{OCR},I_{OCR},F_{OCR}), and\\
&T_{LEX}{=}(\Sigma_{LEX},\Delta_{LEX},Q_{LEX},K_{LEX},E_{LEX},I_{LEX},F_{LEX}).
\end{aligned}
\tag{3}
$$

提案手法では，合成演算を文字認識結果を表現した WFST と語彙の辞書を表現した WFST で行い，両者に共通する状態遷移とその遷移毎の累積重みを求めるために用いている．検出された状態遷移は，より尤もらしい遷移ほど重みが小さい値となり，出力を遷移の重みが小さい順に単語を取り出すことで，一度の WFST の合成演算により文字認識結果の複雑な遷移の組み合わせの中からの語彙検出を実現する．

WFST の実装に当たっては，オープンソースの WFST ライブラリである OpenFST を利用した．

3.1 文字認識結果の WFST

提案手法では，ノイズを含み，切り出し位置が未知の文字列から語彙の検出が行えることを目標としている．そこで，WFST の複雑な状態遷移から最適なパスの選択が行える特性を利用し，複数の OCR エンジンによる処理パターンから大量の認識結果候補を取得する．切り出し位置についてもオーバーセグメンテーションによる切り出し位置の多重仮説を用い，切り出し失敗による検出失敗を回避する．

単純化した文字認識結果の WFST の状態遷移の例を図 6 に示す．オーバーセグメンテーションの切り出し位置から構成される，直列の重みのついた ε 遷移から，1 文字と判定された 2 点の切り出し位置の間に文字認識結果の候補群の遷移が並列に接続された構成となっている．重みのついた ε 遷移は文字のスキップとして機能し，重みは文字認識結果の最も可能性の低い候補と同じ重みとなるように 1.0 としている．また，提案手法では，文字列中の単語の範囲は未知となっているため，例えば”pencil”の一部分から”pen”を検出するというように，単語の途中部分から別の単語が検出される可能性がある．そのため，オーバーセグメンテーションによる切り出し処理にてスペースと判定された二点間には重み 0 のセパレータ記号の遷移も追加する．前述の二点間の ε 遷移の重み は計算式（スペース幅/平均切り出し幅）とした遷移を並列で追加する．こうすることで，同一幅・同一間隔で単語が構成されている場合，重みは一文字をスキップする場合と同等の 1.0 となる．また，短い隙間がスペースとして判定された場合は小さい値となるため，スキップするためのコストを小さくすることができる．

利用する文字認識エンジンの認識結果として，各候補の尤度情報が存在しているが，値としてとられる幅やスケールが一定ではない．そこで，各エンジンからの統合された文字候補の重みは，各エンジン毎の候補順位を利用し，同一文字が候補となる重複数が多いほど重みが小さくなるように設定する．式 (3) に対応する文字認識結果の WFST のパラメータは式 (4) となる．

$$
\begin{aligned}
&\Sigma_{OCR} : \{\text{ 日本語文字 }\}, |\Sigma_{OCR}| \approx 2000,\\
&\Delta_{OCR} := \Sigma_{OCR},\\
&Q_{OCR} : \{\text{ 各文字の左上の位置 }\},\\
&I_{OCR} : \text{文字列の最も左上の位置,}\\
&F_{OCR} : \text{文字列の最も右下の位置,}\\
&\lambda_{OCR}\ and,\ \rho_{OCR} : set\ to\ zero,\ and\\
&E_{OCR} : \{e\}\ with\\
&w(e) = \frac{score(i(e);p(e),n(e))}{\sum score()},
\end{aligned}
$$

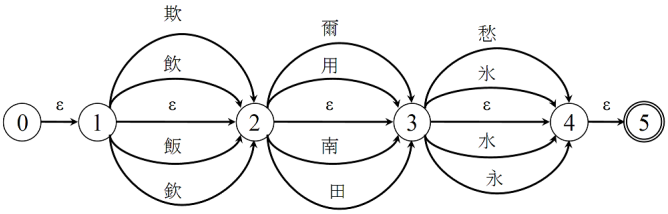


図 6　文字認識結果の WFST の例（「飲用水」の認識結果）

$$where\ score()\ gives\ the\ (1,\cdots,N)th\ ranking\ result\ of\ an\ OCR\ engine\ for\ the\ character\ segment\ (p(e),n(e))\ unless\ i(e)=\{\ \varepsilon\ \}. \tag{4}$$

$$w(e)=1\ ,if\ i(e)=\{\ \varepsilon\ \}. \tag{5}$$

NHOCR による一文字の認識結果として現れる複数文字から構成されるものについては複数文字を分解し，一文字単位の直列の遷移として取り扱う．また，商用エンジンから得られる文字列に対する文字認識の結果は，一文字単位の文字認識の状態遷移と同様に複数の文字候補が連結された状態遷移として表現し，一文字単位の文字認識の状態遷移と並列に接続する．

以上のような構造の状態遷移として文字認識結果を表すことで，認識結果の遷移にノイズが入った状態でも語彙の検出を行うことができ，日本語のような分かち書きでない言語と英語のような分かち書き言語の語彙を識別しつつ同時に検出することを実現する．また，省略された単語についても，重みは大きくなり辞書として登録されている必要があるが，検出することができる．

なお，オーバーセグメンテーションによる重複位置での文字切り出し，複数の一文字認識処理による大量の認識候補の取得を行っているため，文字形状の特性による文字候補の修正は行っていない．また，未知語への対応についても，文字認識の結果を利用することで対応可能だが，情景画像からの文字認識として文章ではなく語彙の検出を目標としており，WFST 処理によるノイズ領域のフィルタリングにも利用しているため行っていない．

3.2　辞書の WFST

語彙の辞書は，図 7 のように，ユニグラムで表現された単語の状態遷移が並列に並んだ構造となっている．また，英語のような分かち書き言語の場合，語彙の始まりと終了位置に，セパレータ記号の遷移を追加する．

式 (3) に対応する辞書の WFST は式 (6) となり，単語長が長く出現頻度が高いほど重みが小さくなるよう設定する．

Σ_{LEX} : { 約１００万の日本語語彙 },
$\Delta_{LEX} := \Sigma_{LEX}$,
Q_{LEX} : 各語彙の文字の論理セグメント,

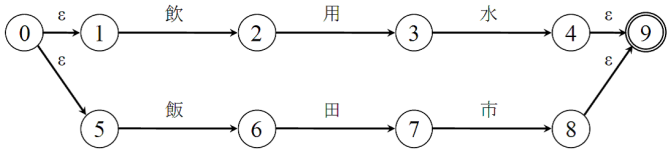


図 7　辞書の WFST の例（単語数 2）

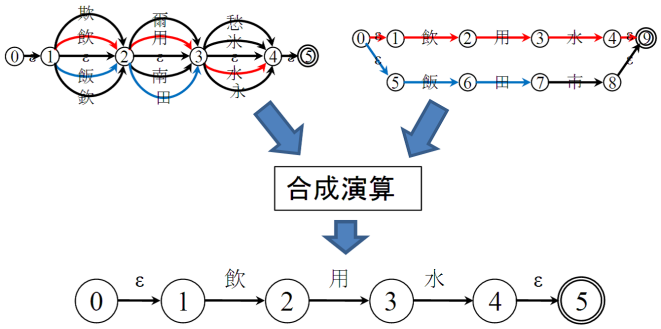


図 8　WFST 合成演算の例

I_{LEX} : 定義された初期状態,
F_{LEX} : 定義された最終状態,
$\lambda_{LEX} and,\ \rho_{LEX} : set\ to\ zero,\ and$
$E_{LEX} : \{e\}\ with$
$w(e) =$
$$\alpha\ (1-\frac{\sqrt{WordLength}}{\sqrt{MaxWordLength}})+\beta\ (1-\frac{\ln(WordFrequency)}{\ln(MaxWordFrequency)}),$$
$if\ (i(e)/o(e))\ spans\ the\ first\ character\ of\ each\ word,$
$else\ w(e)\ is\ set\ to\ zero,\ where\ \alpha+\beta=1.$

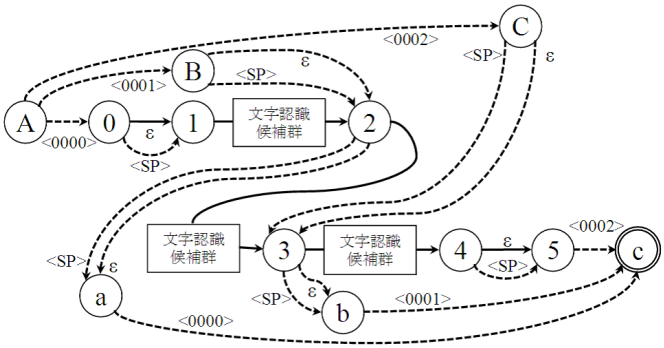
(6)

図 8 に，図 6 の文字認識結果と図 7 の辞書の合成演算の例を示す．辞書には「飲用水」「飯田市」が登録されており，文字認識結果として「飲」「用」「水」の遷移が存在し，「飯」「田」の遷移はあるが，「市」が存在しないため，結果として「飲用水」の遷移のみが残る．また，複数の遷移が残った場合は，重み順に並び替えることで，利用する遷移の優先度を判別することができる．

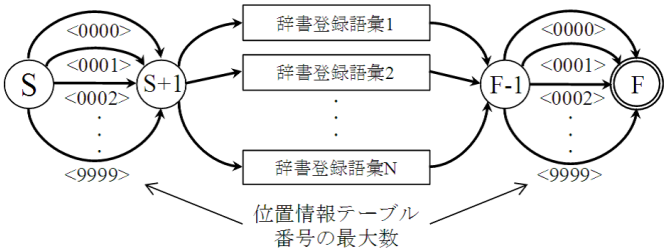
3.3　WFST への位置情報の付加

文字認識結果の WFST としては，各文字認識結果の候補群をつなぐ状態間の遷移について，初期状態から途中の状態，途中の状態から最終状態への ε 遷移を情景画像における位置情報テーブル番号として表現する．また，この遷移は位置情報テーブル番号と，並列化された ε 遷移・セパレータ記号遷移を直列に接続した構成とする．これは辞書において英語の語彙にセパレータ記号を付加しているためで，アルファベットで構成された日本語の場合 ε 遷移が選択され，英語の語彙の場合，セパレータ記号の遷移が選択される．

図 9 に位置情報を付加した文字認識結果の WFST の構造を示す．数字の状態番号が文字認識候補群の遷移を表し先述の図 6 のように文字認識候補群の状態遷移と



(a) 位置情報を付加した文字認識結果の WFST



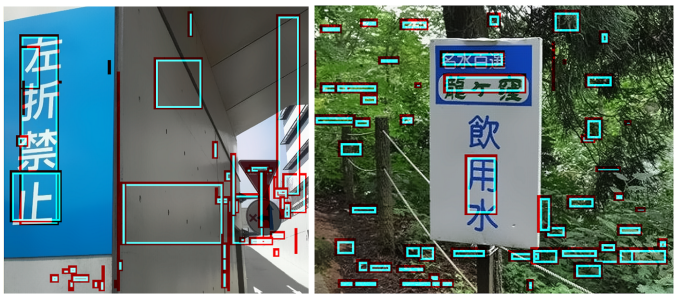
(b) 位置情報を付加した辞書の WFST

< 0000 > ～ < 9999 >：位置情報テーブル番号

< *SP* >：セパレータ記号

ε ：ε 遷移

図 9　位置情報を付加した WFST



(a) ビルの標識　　(b) 森林の看板

図 10　合成演算による語彙検出結果

なっている．また，A～C の状態が文字の開始位置を表した初期状態から途中の状態への遷移，a～c の状態が文字の終了位置を表した途中の状態から最終状態への遷移となっている．

辞書の WFST は，図 7 のように並列に接続された登録語彙の前後に，位置情報テーブル番号として採りうる最大数の番号を付加する．

WFST 合成演算の結果を位置情報テーブルの座標に基づいて出力した結果を図 10 に示す．ノイズの領域の誤検出が現れているが，図 3 で現れていた誤検出領域の多くが除去できている．

4. 実　　験

30 枚の日本語を含む情景画像のデータセットを用意して検出精度についての実験を行った．なお，辞書に含まれる語彙セットは，インターネットからの収集も含め，

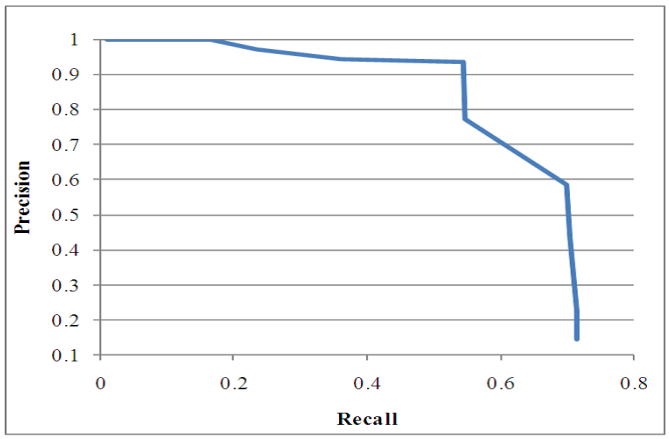


図 11　実験結果の Precision/Recall

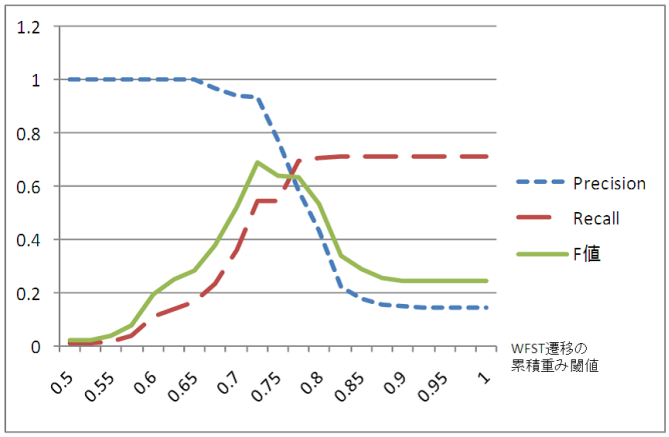


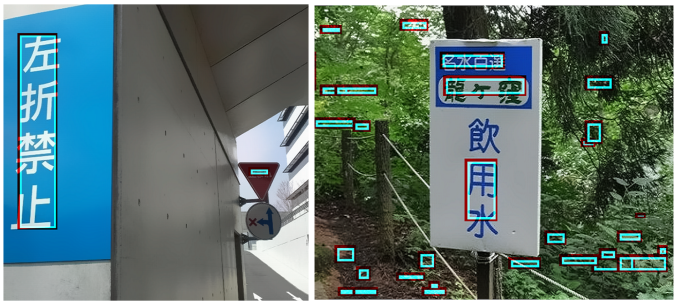
図 12　実験結果の F 値

約 100 万語の語彙を用意した．図 11 に WFST 遷移の累積重みの値を検出閾値とした Precision/Recall の結果を，図 12 に F 値の結果を示す．

実験には 2.8GHz Core 2 processor, 3.2GB RAM (32bit Windows XP single-thread application) を用い，平均処理時間は 145 秒となった．大量の状態と遷移を処理する演算を行っており，網羅的な領域取得のために画像処理としての効率化を行っていない状態であるが，処理時間と精度の面で実用的な性能を実現できるものと考える．

図 10 において閾値を 0.75 未満にしたときの語彙検出結果を図 13 に示す．ビルの標識においては正解語彙のみを検出する良好な結果を得られているが，森林の看板については正解語彙のほか，木の葉の領域に誤検出が残っている．誤検出された語彙を見ると「ニニニニ」といった単純な形状の文字で構成された語彙が検出されていた．辞書については，インターネット等から収集された語彙も含まれているため，出現頻度が低くノイズとして現れやすい語彙も多く，今後はこれらの語彙の重みについて検討する必要がある．

図 14 にその他の情景画像における語彙と領域の検出結果例を示す．



(a) ビルの標識　　(b) 森林の看板

図 13　重みの閾値による絞込み結果

5. 結　　論

情景画像中からの日本語語彙検出手法として，文字認識結果と大規模語彙による辞書を位置情報を保持する特性を持つ WFST であらわし，WFST の演算により語彙を検出する手法の提案を行った．実験により，大量のノイズを含む候補領域の中から，文字認識結果として語彙の検出が行えることが確認できた．

画像処理については，候補領域の判定の品質より網羅的に領域を取得することを目標としていたこともあり，処理速度が課題となるが，Adaboost 等の手法を用いた候補領域検出の精査を行うことで改善が図れるものと考える．言語処理についても，現状では単純なユニグラムによる辞書を用いており，N グラム等のより高度な言語モデルを用いることでの精度向上が期待できると考える．WFST の文字認識結果の重み，辞書の重みについても，日本語語彙と英語語彙の文字の特性の差異を考慮し，今後更なる検討を進め精度向上を図る予定である．また，WFST の処理手順を見直すことでの高速化も検討していく予定である．

文　　献

[1] Welcome to your notable world: Evernote Corporation, http://www.evernote.com/ .

[2] L. P. Sosa, S.M. Lucas, A. Panoretos, L. Sosa, A. Tang, S. Wong, and R. Youg, K. Ashida, H. Nagai, M. Okamoto, H. Yamamoto, H. Miyao, J.Zhu, W.Ou, C. Wolf, J-M Jolion, L. Tooran, M. Worring, X. Lin, ICDAR 2003 Robust Reading Competitions, IJDAR, vol.7, pp.105-122, 2005.

[3] S.M. Lucas, ICDAR 2005 text locating competition, Proc. ICDAR '05, pp.80-84, 2005.

[4] Y. Pan, X. Hou, C. Liu, A robust system to detect and localize texts in natural scene images, Proc. DAS'08, pp.35-42, 2008.

[5] Y. Kusachi, A. Suzuki, N. Ito, and K., Arakawa, Pattern Recognition, Kanji recognition in scene images without detection of text fields-robust against variation of viewpoint, contrast, and background texture, vol. 1, pp. 457-460, 2004.

[6] L. Xu, H. Nagayoshi, and H.Sako, Kanji character detection from complex real scene images based on character properties, Proc. 8th Int. Workshop on Document Analysis Systems, pp.278-285, 2008.

[7] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri, OpenFst: a general and efficient weighted finite-state transducer library, Proc. of the CIAA'07, pp. 11-23, 2007.

[8] K. Zhu, F. Qi, R. Jiang, L. Xu, Using Adaboost to Detect and Segment Characters from Natural Scenes, Proc. CBDAR 2005, pp.52-59.

[9] T. Retornaz and B. Marcotegui, Scene text localization based on the ultimate opening, Proc. Int. Symposium on Mathematical Morphology, pp.177-188, 2007.

[10] S. Messelodi and C. M. Modena, Automatic identification and skew estimation of text linesin real scene images, Pattern Recognition, vol. 32, pp 791-810, 1999.

[11] R. Casey and E. Lecolinet, A survey of methods and strategies in character segmentation, IEEE Trans. on PAMI, vol.18, no.7, pp.690-706, 1996.

[12] NHOCR, http://code.google.com/p/nhocr/. [Online].

[13] R. Beaufort, and C. Mancas-Thillou, A weighted finite-state framework for correcting errors in natural scene OCR, Proc. ICDAR '07, pp.889-893, 2007.

[14] Z. Saidane, C. Garcia, and J.L. Dugelay, The image text recognition graph (iTRG), Proc. ICME 2009, pp.266-269, 2009.

[15] T. Breuel, The OCRopus open source OCR system, Proc. IS&T/SPIE 20th Annual Symp., 2008.

[16] Graham NEUBIG, 森 信介, 河原 達也,"重み付き有限状態トランスデューサーを用いた文字誤り訂正",",言語処理学会第十五回年次大会 発表論文集",pp.332-335,March 2009.

(a) ロゴマーク　　(b) 街中の看板

(c) 商品名　　(d) 駐車場案内

図 14　処理結果例