ジオリファレンス情報を用いた空間情報抽出システム

相良毅[†1] 有川正俊[†1] 坂内正夫[†2]

WWW ページや電子メールをはじめとするネットワーク上の様々なデータには，ジオリファレンス記述を含むデータが多数流通している．これらのデータに含まれる地名や住所などの情報を効率よく取り出し，緯度経度のような空間的位置座標に変換すると，ナビゲーションシステムなどの多くの空間情報アプリケーションで有効に利用できる．

本稿では，空間情報に特化した日本語自然言語処理による半構造化処理と，アドレスマッチング手法を用いて，ネットワーク上を流通する情報に対してこのような変換処理を行う空間情報抽出システムを提案し，構成とアルゴリズムを説明する．また，実システムの一例として空間情報サーチエンジンを作成し，空間情報抽出システムの有効性を確認したので報告を行う．

Spatial Information Extraction System using Geo-referenced Information

TAKESHI SAGARA,[†1] MASATOSHI ARIKAWA[†1] and MASAO SAKAUCHI[†2]

Today, a large amount of various data such as web pages and email messages are exchanged on the Internet. Most of the data include geo-referenced descriptions such as addresses and names of buildings. Once geo-referenced descriptions are extracted from the data on the Internet and converted to pieces of geographic coordination, that is longitude and latitude, the data become very useful for spatial information applications like navigation systems.

The systems supporting these functions are named "spatial information extraction systems", because spatial descriptions can integrate most of various data on the Internet as spatial data. This paper introduces basic algorithms for the systems and shows some results of evaluating the algorithms. We also present some demonstrations of our developed system "Basho" based on these algorithms.

1. はじめに

地理情報システム（Geographic Information System:GIS）[1]は，施設管理や顧客管理，交通管理，経営戦略支援など，空間的に情報が分布するデータの管理と解析のためのツールとして，徐々に普及している．しかし，地図データや関連するデータの作成・調査に多くのコストがかかるため，主に地方自治体や大企業などに利用が限られているのが実状である．

一方で，コンピュータのハードウェア面の著しい進歩により，低額の PC でも，地図データの幾何演算や検索に必要な計算能力と，データの管理に必要な記憶容量を持つようになっている．そのため，カーナビゲーションシステムや PC 用地図ソフトウェアなど，個人レベルで利用できる GIS の需要が高まっている．さらに最近では，携帯端末によるモバイル技術を利用したパーソナルナビゲーションも注目を集めている[2]．しかし，個人で利用できる地図データや，地図に重ねあわせるコンテンツデータは作成に依然コストがかかるため，データ不足が問題となっている．

我々は，インターネット上のWWWページや電子メールに多くの地理参照情報が含まれていることに着目し，これらの情報源から地名や住所のような地理情報記述を抽出し，GIS で利用できる形式に変換する手法を開発した．また，本手法を実装した空間情報抽出システム『芭蕉』を開発し，サンプルアプリケーションとして空間サーチエンジンシステムを作成した．

本システムは，まず WWW ページなどに含まれる自然言語文章から半構造化手法によって地名や住所の部分を抜き出し，我々が<spa>表現と呼ぶ XML 記述を利用した一種の中間表現を生成する．次に，アドレスマッチング手法により，抜き出した地名を緯度経度表現に変換し，より精度の高い<spa>表現を生成する．

以下，まず 2 節で本手法の目的を明確化するために対象とするデータの分類を行い，3 節で<spa>表現を提案する．次に，4 節でアドレスマッチング手法を，5 節で自然言語処理を用いた半構造化手法について説明する．6 節では提案する手法の評価を行い，最後に，7 節で提案手法の有効性を示す例として実装した空間サーチエンジンシステムを紹介する．

†1 東京大学空間情報科学研究センター
Center for Spatial Information Science at University of Tokyo
†2 東京大学生産技術研究所
Institute of Industrial Science, University of Tokyo

2. 空間データの分類

従来の GIS で利用可能なデータには，地理データ（Geographic Data）と地理参照データ（Geo-referenced Data）がある．地理データは道路形状や行政界などの幾何的な情報を含むデータで，地図データに ID や名称などの基礎的関連情報が付属したものである．地理参照データは，顧客伝票や道路の交通量，市町村人口などの定量データに，地理データと結合するための住所や市町村名などの地理参照データを加えたものである（**図 1**）．目的にあった地理データと地理参照データを結合することにより，管理や解析といった用途に用いることができる．これらのデータはそもそも特定目的用に多くの費用をかけて作成するものであり，高い精度が要求される．

さて，日常生活で何気なく交わされる情報には，待ち合わせ場所の指定など，住所や地名などを含むものも多い．そこで地理データよりも広義の概念として「空間的な位置情報を含むデータ」を「空間データ（Spatial Data）」と定義する．空間データには，「○△町□番地で火災発生」「震源地は××沖 50km」や「○○駅前の△ラーメンはおいしい」といった自然言語で記述された文章や，略地図，事故現場を写すニュース映像なども含むことができる．

これらの高級な表現は人間にとっては有用だが，そのままではコンピュータには理解できないため，利用が難しい．このような表現の曖昧さを解消し，検索などの利用を可能にする手法として，近年 XML[3]などの半構造化表現を利用したドキュメント記述が非常に注目されている[4),5)]．たとえば「○△町□番地で火災発生」というデータを「<spatial information> <location> ○△町□番地 </location> で <phenomena> 火災発生 </phenomena> </spatial information>」のように記述すれば，コンピュータにとって理解可能になる．しかもデータ作成者の負担は，スキーマが固定である構造化データを作るよりも少なく，個人レベルでのデータ発信や流通に利用することができる．XML による空間データの記述の詳細については 3 節で述べる．

空間データは，構造化のレベルと空間位置の記述方法によって分類することができる（**図 2**）．まず，自然言語や画像などの表現を用いた空間データを非構造化データ(Non-structured data)，XML などの構造化文書表現を利用したものを半構造化データ(Semi-structured data)，地理参照データや地図データのように特定のフォーマットにしたがったものを構造化データ(Full-structured data)と分類する．別の分類法として，地理データのように位置を座標値で記述したものを直接空間参照データ（Directly spatial referenced data），住所などの地理参照記述を用いたものを間接空間参照データ（Indirectly spatial referenced data）と分類する．両者の分類法を組み合わせると，空間情報を 6 種類に分類することができる．以下ではそれぞれの頭文字を取り，構造化-直接参照データを F-D データ(Full-structured, Directly spatial referenced)，非構造化-間接参照データを N-I データ（Non-structured, Indirectly spatial referenced）のように表記する．

さて，6 種類の空間データのうち，コンピュータにとってそのまま地図上に投影して利用することが可能なのは F-D データのみである．WWW ページや電子メールなど，ネットワーク上を流通するデータを有効に利用するには，それ以外の 5 種類のデータを F-D データに変換する手法が必要である．半構造化データから構造化データを作成するのは一般的な XML パーサを利用すれば可能なので，間接参照データから直接参照データを作成する手法，および，非構造化データから半構造化データを作成するための手法が必要である．

3. XML による空間データ記述

従来，GIS で利用する地理データは，市販されている基盤データを購入したり，測量やディジタイズによ

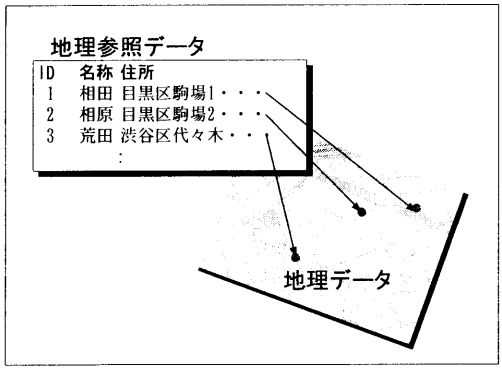

図 1 地理データと地理参照データ
Fig. 1 Geographic data and Geo-referenced data

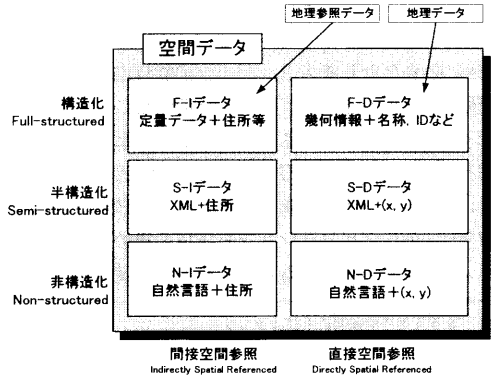

図 2 空間データの分類
Fig. 2 Classification of Spatial Data

って作成する必要があり，コストの高さが問題になっている．そこで，過去に作成した地理データの流通交換を進めてGISの相互操作性を高め，地理データ記述を標準化するためにXMLを利用しようという動きがある．

GISベンダーを中心に構成されているOpen GIS Consortium（OGC）では，F-Dデータである地理データをXMLで記述して共通で利用するためのXMLフォーマットを検討している[6]．また国内でも，XML記述を利用した地理データ交換標準としてG-XMLが検討されている[7]．「G-XMLプロジェクトは，GISコンテンツが利用者側の電子地図やGISエンジン等に左右されず自由にインターネット上を流通し，さらにデータの検索や加工等も容易に行えるようにすることを最終目的とした」もので，OGCよりも広いS-I，S-D，F-Dデータを対象としている．

我々はこれらの動きと並行して，個人レベルでの空間データ発信・流通のための空間情報XML記述，「<spa>表現」を提案する．<spa>表現の主な目的は，GIS専門家ではない一般利用者でも記述しやすいように表記が容易であることと，直接参照だけではなく間接参照による空間情報記述も許可することの2点であり，主にS-I，S-Dデータを対象とする．最もシンプルな<spa>表現は次のようになる．

```
<spa name="渋谷"/>
```

場所を表す表現は，その性質上，他の文書中に埋め込まれることが多いため，<spa>表現はXMLの文書要素(DOCTYPE)ではなく，namespaceにより他の文書要素中の一要素として利用されることを想定している．例えば，会議の案内文書で会場の場所を説明する部分を<spa>と</spa>で囲むことで，元の文章に影響を与えることなく場所を表す部分を記述することができる．より詳細な表現が可能な<spa>表現の完全なDTDを図3に示す．次の例を用いて各要素を説明する．

```
・・・次回の会議は
<spa name="東京大学空間情報科学研究センター
会議室" type="point" p="139.83E 35.614N">
  東京大学空間情報科学研究センター会議室
  <geoword>東京大学</geoword>
  <keyword>会議</keyword>
  <author>相良 毅</author>
</spa>で×月○日△時より・・・
```

<spa>内のname属性はこのデータの名称であり，任意の文字列を指定できる．例では，これから説明する空間情報が「東京大学空間情報科学研究センター会議室」に関するものであり，type属性で形状が点（point）であることを表す．p属性は直接参照データの座標値で，緯度経度が139.83E，35.614Nであることを示している．次のgeoword要素は省略可能で，地理参照を記述する．p属性が指定されていない間接参照データの場合はgeoword要素を調べることで，おおよその位置を知ることができる．つまり，p属性が記述されていればS-Dデータ，geoword要素のみ記述されている場合はS-Iデータである．

author要素はこのデータを作成・入力した著者や生成したプログラム名を入力する．keyword要素には，このデータを検索する場合のキーワードを記述する．

4. アドレスマッチング

地理データと地理参照データを結合するには，地理参照データに含まれている住所や郵便番号などの地理参照記述をキーとして，地理データ上の領域や点に対応づける処理が必要である．この処理をアドレスマッチング（Address Matching）と呼ぶ[8]．アドレスマッチングを行えば，顧客データに含まれる住所情報から顧客の分散地図を作成することが可能になるなど，メリットが大きく，非常に重要な技術である．

アドレスマッチングは原理的には（地名：座標）の参照テーブルを作成し，地名を検索して座標値を返す（図4）．地理参照記述には地名の他に電話番号や郵便番号，建物名なども利用できるが，原理は同一である．GISソフトウェアによっては，地理データから参照テーブルを自動作成してアドレスマッチングを行う

```
<!ELEMENT spa (#PCDATA | geoword? | keyword? | author?) >
<!ATTLIST spa name CDATA    #IMPLIED
          type (point | line | area) "point" #IMPLIED
          p    NMTOKENS  #IMPLIED >
<!ELEMENT geoword (#PCDATA) >
<!ELEMENT keyword (#PCDATA) >
<!ELEMENT author (#PCDATA) >
```

図3 <spa>表現のDTD
Fig.3 <spa> representation DTD

機能を持つものもある[10]．しかし，地理データを用意した範囲内しかアドレスマッチングできないため，広い範囲に対するアドレスマッチングを行おうとすると多くの地理データが必要となりコスト面で問題になる．

アドレスマッチングを空間データ全般に拡張すると，間接空間参照データを直接空間参照データに変換する手法と捉えることができる．しかし，既存の地理参照データに対するアドレスマッチング手法では，たとえば第1フィールドに都道府県名，第2フィールドに市町村名を記述するといった構造化された表記を前提にしていたり，数字はすべて全角アラビア表記とする，などといった制約がある．一方，WWW ページなどで一般的に用いられている間接参照データは，市町村名が省略されていたり，漢数字・アラビア数字が混在しているなど表現にばらつきがあり，そのままでは利用が困難である．

また，ネットワーク上の空間データには全国の地名が含まれているため，これらの地理参照表現を座標値に変換するためには日本全国をカバーする必要がある．そのため既存の GIS によるアドレスマッチングでは，日本全国の詳細な地理データを用意しなければならず，データ量とコストの面で現実的ではない．そこで，地理データを別に用意せずに，かつ，ばらつきのある表現でも安定してアドレスマッチングを行う手法を開発した[11]．

上で述べた通り，アドレスマッチングでは地名と座標の対応表から地名を検索し，座標値を得るため，対応表のデータ構造が検索効率やメモリ消費量に大きく影響する．地名は多くの場合階層構造を持っているので（東京大学／空間情報科学研究センター，東京都／目黒区，小田急線／下北沢駅など），階層木構造を利用することで検索効率を上げ，メモリ消費量を減らすことができる．この階層木構造をこれ以降「地名管理木」と呼ぶ．地名管理木の各ノードは「目黒区」などの地名に対応し，その地名が表す領域の代表点座標を持つ．また，空間的な階層にしたがった上位ノード（「東京都」）と下位ノード群（「駒場1丁目」「駒場2丁目」…）へのリンクも管理する．

日常的な会話や文章で地名を表現する際には上位階層を省略する場合が多く，地名管理木の下位ノードからでも直接検索できる必要がある．そこで，地名管理木の各ノードを葉ノードとする TRY 構造を用いた二重管理構造を利用する（図5）．この TRY 構造をこれ以降「地名インデックス」と呼ぶ．二重管理構造を利用して地名を検索する手順を以下に示す．

(1) 地名インデックスを利用して，文字列の先頭から一致文字数が最長となるノード n_k を見つける．

(2) ノード n_k の子ノード集合 $\{n_{(k,1)}, n_{(k,2)}, ..n_{(k,m)}\}$（$m$ は子ノード数）から，文字列の続きの部分が一致するものを探し，親ノードを含めた一致文字数が最長となるノードを見つけ，n_{k+1} とする．

(3) (2)を再帰的に繰り返し，最長一致ノード系列 $\{n_k, n_{k+1}, ..., n_{k+i-1}\}$（$i$ はノード列長）を得る

(4) n_k の親ノードを上に辿り，上位階層も含んだ完全なノード系列 $\{n_{k-j}, n_{k-j+1}, ..., n_k, ..., n_{k+i-1}\}$（$j$ は n_1 から最上位ノードまでの中間ノード数）を得る
((1)～(3)において，最長一致解が複数ある場合はその全てについて同じ処理を行う)

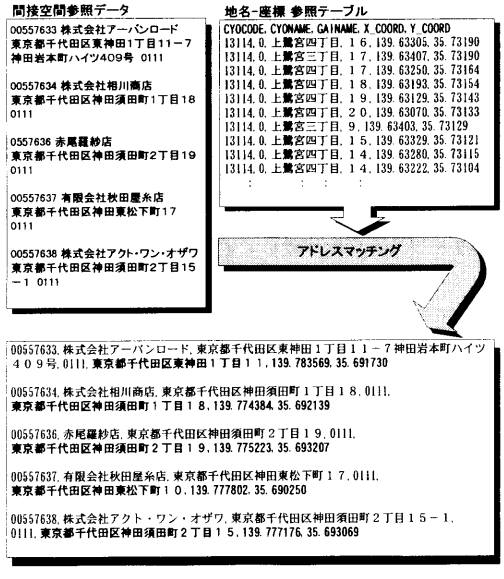

図4 アドレスマッチング
Fig.4 An example of address matching

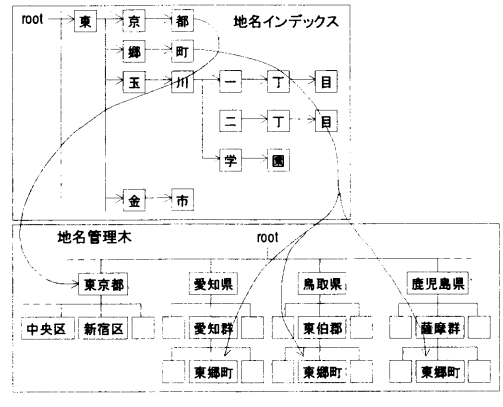

図5 アドレスマッチング用 二重管理構造
Fig 5. Two tree structures for proposed address matching method

地名には（東京都）中央区と（名古屋市）中央区のように同名のものがあるため，解が複数存在することがある．そこで，一致度に応じて次のような得点付けを行い，得点が高いものから採用する．

score=1: 文字列の先頭部分に部分一致する地名だけが存在する(例:「渋谷区役所」を問い合わせたが，対応表には「渋谷駅」「渋谷警察署」しかない)

score=2: 文字列の先頭部分に完全一致する地名が存在するが，同じ地名が複数存在する(例:「南台一丁目」を問い合わせたが，対応表には「東京都/渋谷区/南台一丁目」と「埼玉県/川越市/南台一丁目」が存在する)

score=3: 文字列の先頭部分に完全一致する地名がただ一つ存在する(例:「駒場四丁目」を問い合わせ，「東京都/目黒区/駒場四丁目」だけが一致した)

score=4: 文字列の先頭部分に完全一致する地名が存在し，文字列の後続部分に一致する地名も存在する．かつ，両者には地理的な親子関係(県－市，市－区など)が成立している(例:「目黒区駒場四丁目」を問い合わせ，「目黒区」と「駒場四丁目」が辞書と一致した．かつ，「目黒区」内には「駒場四丁目」が存在する)

用途によってはscore=2以上で有用なこともあるが，score=4以上でなければ十分な利用価値がない場合もある（7.2参照）．

内容	備考	件数
全国の市町村名		4,118件
住所データ	東京都，埼玉県，千葉県，神奈川県	618,156件 重複あり
首都圏駅名	数値地図2500より	1,272件
首都圏建物名	数値地図2500より	37,803件
交差点名，駅名，高速道路インターチェンジ名	全国デジタル道路地図中より	24,921件

表1 アドレスマッチングテーブル登録済みデータ
Table.1 Registered data in an address matching table

以上の処理により，間接空間参照データを効率良く直接変換参照データに変換することができる．この変換処理の効率に関しては，6節で詳しく説明する．

5. 日本語自然言語処理による半構造化手法

ネットワーク上に存在する空間データはほとんどがN-I データであり，これらのデータの利用が重要である．そこで，N-Iデータから<spa>表現を利用したS-Iデータを作成する手法を開発した．

5.1 空間情報に特化した単語切り出し処理

WWWページなどのN-Iデータから文章を取り出し，形態素解析システム『茶筅』[12]を利用して品詞分解を行い，分解された単語列から「地名らしい」ものを選択する．理想的な形態素解析システムでは，正しく解析が行われれば全ての地名は固有名詞として分解されるはずである．しかし，実際の形態素解析システムでは，単語辞書を利用して「もっとも連続性の高い品詞列の組み合わせ」を求めるため，辞書に登録されていない地名は正しく切り出されない．

あらゆる地名をすべて辞書に登録することは非現実的であり，一般名詞と区別できない地名もあるので，全ての地名を正しく固有名詞として分解することは不可能である．しかし，多くの場合，地名は複合名詞になっているため，複数の連続する名詞を抽出し，地名のみの辞書を用いて検証することで地名を抽出することがある程度可能になる．「地名らしい」単語列の選択基準ルール（以下，地名選択ルールと呼ぶ）を示す．

「グループ1に含まれる品詞から始まる単語列であり」，かつ「グループ1またはグループ2に含まれる単語が0回以上連続していること」

グループ1の品詞
名詞・固有名詞・一般，名詞・固有名詞・人名・一般，名詞・固有名詞・人名・姓，名詞・固有名詞・一般・名，名詞・固有名詞・組織，名詞・固有名詞・地域・一般

グループ2の品詞
名詞・一般，名詞・固有名詞・人名・名，名詞・固有名詞・国，名詞・サ変接続，名詞・数，名詞・接尾・一般
(品詞区分は『茶筅』に従う)

地名選択ルールは，表1に示したアドレスマッチング用の地名辞書に登録してある地名に対して形態素解析を行い，解析結果の和集合となる最小の組み合わせである．グループ1，2に他の品詞を加えれば，それだけ地名を拾う可能性は高くなる(再現率が高くなる)が，地名以外の文字列を拾う可能性も高くなり（適合

率が低くなる），処理速度が低下する．そのためアドレスマッチング辞書を拡張する場合は，辞書にあわせてルールを調整する必要がある．

5.2 <spa>表現埋め込みのための半構造化処理

次に，地理参照記述を含む日本語文章から，<spa>表現を含む半構造化ドキュメントを作成する手順を説明する．

(1) 日本語文章を文の列$\{s_1, s_2, \ldots, s_n\}$($n$は文の数)に分解する．

(2) i=1から始め，i番目の文s_iを形態素解析し，単語列$\{w_{(i,1)}, w_{(i,2)}, \ldots w_{(i,m)}\}$($m$は$s_i$に含まれる単語数)を作成する．

(3) $w_{(i,k)}$をk=1から始め，地名選択ルールを利用して地名かどうか判定する．地名であると判断された場合(4)以降の処理を，地名ではないと判断された場合(6)の処理を行う．

(4) (3)で$w_{(i,k)}$が地名であると判定された場合，$w_{(i,k)}$と$w_{(i,k+1)}$を文字列として結合し，新たな単語$w_{(i,k\cup(k+1))}$を作成し，再び地名選択ルールを適用する．地名であると判定される限り文字列を結合し，最長となる単語$w_{(i,k\cup(k+1)\cup\ldots\cup(k+j))}$($j$は結合した単語数)を得る．

(5) 得られた単語を<spa>表現に置き換え，「<spa name=" $w_{(i,k\cup(k+1)\cup\ldots\cup(k+j))}$ "> $w_{(i,k\cup(k+1)\cup\ldots\cup(k+j))}$ </spa>」を結果に出力する．kをj増やし，次の地名を探すために(3)から繰り返す．

(6) (3)で$w_{(i,k)}$が地名でないと判定された場合はそのまま結果に出力する．また，kを1増やし(3)の処理を繰り返して次の単語を調べる．$k=m$となった場合はiを1増やし，(2)から繰り返して次の文を調べる．

以上の処理により，自然言語文章に<spa>表現を埋め込んだ半構造化文章を作成することができる（例1）．

池ノ上駅前のパン屋のゴマ餡パンはお勧めです。

↓

<spa name="池ノ上駅">池ノ上駅</spa>前のパン屋のゴマ餡パンはお勧めです。

例1 半構造化処理の例

5.3 構文化文書に対する前処理

5.1で述べた自然言語処理の結果は，形態素解析の結果に大きく依存する．しかし，そもそも形態素解析は自然言語で書かれた日本語を対象としているため，HTML文書のようにタグで細かく切断された文章を解析すると精度が大きく低下する原因になる．そこで，前処理を行って入力となるN-Iデータをなるべく平易な日本語に変換しておく必要がある．

HTML文書の場合の前処理は，全てのHTMLタグを取り除けば良い．ただし，HTMLタグ中にも地名が含まれていることがあるため，タグ内の文も別に処理する必要がある．

6. 提案手法の評価

本節では，4節で提案したアドレスマッチング手法および5節で提案した構文解析手法の評価実験およびその結果を報告する．

6.1 アドレスマッチング手法の評価

提案したアドレスマッチング手法は一種の探索木なので，地名参照テーブルに登録されている地名であれば100%発見できる．そこで，アドレスマッチングを行う際の地名の検索効率について考察する．

一件あたりの検索時間をT_{Total}とすると，そのうち地名インデックスによる検索時間T_{TRY}と地名管理木による検索時間T_{Tree}は次の関係がある．

$$T_{Total} = T_{TRY} + T_{Tree} \quad \ldots\ldots(1)$$

地名インデックスは，検索する地名語の文字数がαである場合，α段の階層構造をたどる．k段目のノード数をm_kとすると，検索時間T_{TRY}は次式で表される．

$$T_{TRY} = \Sigma_{(k=1\ldots\alpha)}(C \log m_k)\ (C\text{: 定数}) \quad \ldots\ldots(2)$$

各ノードの子ノード数m_kを一定（$=m$）と仮定すれば，T_{TRY}は次式で表せる．

$$T_{TRY} = C\alpha \log m \quad \ldots\ldots(3)$$

実際にはmは一定ではなく，ある文字の後には特定の文字がくる確率が高いという言語固有の偏りがある．そこで数値地図2500に含まれる東京都・埼玉県・千葉県・神奈川県それぞれの行政界と，駅名，建物名について個別に地名インデックスを作成し，mの偏りを調査した（表2）．その結果，いずれの場合も$m=2$となる確率が70%以上，$m=1$となる確率が10%以上であり，$m=1$または2となる確率は90%以上であった．つまり，mの値はデータの種類や数に依存せず，(3)よりT_{TRY}は地名語の文字数に比例する．

次に，地名管理木ににについて考察を行う．地名管理木のノード数は分散が大きい分布になる．例えば行政区を木構造で管理する場合，各ノードの子ノード数は

下位行政区（県ノードであれば市・郡）の数であり，全国の市長村と東京都・埼玉県・千葉県・神奈川県の行政区を木構造にした場合，子ノード数は**図 6** のような分布になる（葉ノードは含んでいない）．図からもわかるとおり，100 以上のノードを持つものも多い．しかし，子ノード数は地名探索木中の全ノード数 N とは無関係である．

一方，検索の際にたどる階層の数は，検索する地名語が何段階の地名語から構成されているかによって決定される．例えば「東京都目黒区駒場 4 丁目」を検索する場合は「東京都」「目黒区」「駒場 4 丁目」の 3 段階であり，3 つのノードをたどる．同じ内容の地名でも「目黒区駒場 4 丁目」であれば 2 つのノードしかたどらない．つまり，検索時にたどる階層の数も全ノード数 N とは無関係で，地名語の表記に依存する．

以上の考察より，T_{Tree} は地名語の表記に依存し，地名参照テーブルの大きさには依存しない．よって(1)より，T_{Total} は地名参照テーブルの大きさとは無関係であるといえる．そこで次のような実験を行い，検証を行った．

	0	1	2〜8	9+
東京	17.35	77.07	5.28	0.29
埼玉	18.94	74.60	6.03	0.42
千葉	20.85	72.92	5.55	0.67
神奈川	18.93	74.67	5.99	0.40
建物	11.21	84.40	4.13	0.26
駅	20.63	72.35	6.31	0.70

表 2 地名インデックスの子ノード数分布（%）
Table.2 Distribution of the number of child nodes in the TRY index structure

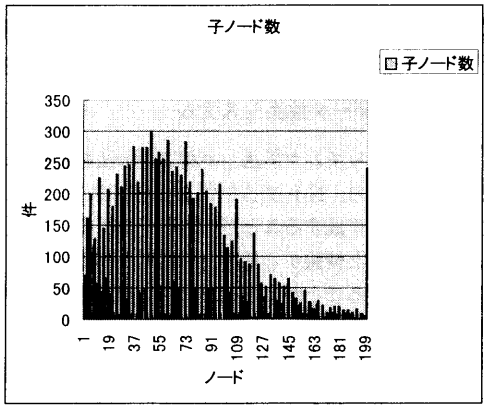

図 6 地名管理木のノード数分布
Fig.6 Distribution of the number of child nodes in the hierarchical tree structure

実験 1. 千代田区内の住所を含む 100 件の質問データをアドレスマッチングするのにかかる時間を，地名参照テーブルの大きさと，住所表記を変えて測定した．利用した地名参照テーブルは，数値地図 2500 から作成したもので，次の 3 種類である(括弧内はノード数)．

テーブル 1)千代田区行政界(1,272)
テーブル 2)東京都行政界(123,763)
テーブル 3)表 1 に示した全データ(686,270)

また，質問データの住所表記は次の 3 種類である．

データ 1)「東京都」から表記したもの(東京都千代田区 xx 町・・・)
データ 2)「千代田区」から表記したもの(千代田区 xx 町・・・)
データ 3)町名から表記したもの(xx 町・・・)

実験の結果を**表 3** に示す．この結果では，地名参照テーブルが大きくなるとわずかに検索時間が長くなっているが，住所表記の違いによる検索時間の変化が支配的であることが分かる．なお，（テーブル 1，データ 1）の組み合わせが（テーブル 2，データ 1）の組み合わせに比べ短時間で処理が終わっているが，これはテーブル 1 には千代田区以外のデータが入っていないので，「東京都」の子ノードが 1 つしかなく，実質的にデータ 2 による検索と変わらないためである．

この結果から，提案した手法における検索時間は地名参照テーブルの大きさに依存せず，地名表現を省略するほど高速になることが確認できた．ただし，地名表現を省略すると，同名の地名があった場合にはどちらかを特定することができないという問題がある．

また，今回実装した計算機は PentiumII300MHz の PC 上に Linux を OS とした載せたもので，特に高度なハードウェアではないが，検索 1 件あたり 0.05 秒以内と十分実用に耐える性能を示した．

6.2 既存手法との比較

商用の GIS ソフトウェアに実装されているアドレ

	データ 1	データ 2	データ 3
テーブル 1	2.20	2.10	0.47
テーブル 2	4.33	2.36	0.50
テーブル 3	4.37	2.36	0.53

表 3 100 件の質問データのアドレスマッチング所要時間(秒)
Table.3 Processing time for 100 queries of address matching (seconds)

スマッチングのアルゴリズムは公開されていないため，ここでは比較対象として 国内で広く利用されているGIS ソフトである ESRI 社の ArcView におけるアドレスマッチングと比較する．ArcView にはアドレスマッチングを行う機能が含まれているが，米国の住所体系と日本の住所体系が異なるため，日本では利用できない．代替的に利用されている方法は次の通りである．

(1) 住所と緯度・経度をもつアドレスマッチングテーブルを用意する．

(2) アドレスマッチングを行いたいデータテーブルに含まれる住所をキーとして，用意したアドレスマッチングテーブルと結合処理を行う．

この方法は GIS ソフトウェアに限らず，一般的なDBMS でも行うことができる．しかし，この方法では，データテーブルとアドレスマッチングテーブルの住所が，文字列として比較した場合に完全に同じでなければ結合できない．そのため，どちらかのテーブルの一方で都道府県名が省略されているなど場合には結合できない（問題 1）．

また，一般的なアドレスマッチングテーブルでは，都道府県名，市区町村名，街区名（大字），番（字），号がそれぞれ別のフィールドに記述されていることが多い．これは，データテーブルで都道府県名や市区町村名が別のフィールドに分かれていれば，まず都道府県名で結合し，次に市区町村名を結合することで検索対象を絞り込むことができ，処理が高速に行えるためである．しかし，データテーブル側で都道府県名や市区町村名が個別のフィールドになっていない場合，前処理としてアドレスマッチングテーブルのスキーマにあわせてこれらを切り分ける処理が必要になる（問題 2）．

さらに，全国を詳細にカバーする大規模なテーブルが仮に存在したとしても，アドレスマッチングテーブルが大きくなるほど結合処理に時間がかかるため，実用的ではない（問題 3）．

提案したアドレスマッチング手法では，都道府県名や市区町村名が省略されていても，中間ノードから検索を開始できるため，問題 1 を解決できる．また，都道府県名や市町村名が分けられていなくても，最長一致となる文字列を探す過程で自動的に切り分けることができるため問題 2 も解決する．WWW 上の地名のように表現にばらつきがあるデータを処理する場合には，提案手法の入力データに対する柔軟性が重要である．また，5.1 で示した通りアドレスマッチングテーブルが大きくなっても性能がほとんど変わらないため，問題 3 も解決できる．

6.3 半構造化処理の評価

半構造化処理では，日本語文章から地名を検出する適合率および再現率が重要である．そこで，次の実験を行った．

実験 2. 実際の WWW ページから提案手法によって地名を抽出し，人手による検査を行い再現率，適合率を求めた．なお，実験に利用したページは「東京のラーメン屋さん[13]」(サンプル 1)及び「全国温泉案内『Oh！YU』[14]」(サンプル 2)からそれぞれ 10 ページずつ任意に抜き出したものである．この実験では，適合率と再現率を次のように定義した．まず，抽出された文字列のうち，人間が見て地名を含むと判断したものの数を N_{good} とし，地名を含まないと判断したものの数を N_{false} とする．また，元の文章から人手で抽出した地名の数を N_{total} とする．

適合率 $P_{precision} = N_{good} / (N_{good} + N_{false})$

再現率 $P_{recall} = N_{good} / N_{total}$

実験の結果を**表 4** に示す．サンプル 1 には 452 件の地名が含まれており，そのうち「小滝橋通り」を除くすべて抽出された．サンプル 2 では 92 件中全件が抽出された．「小滝橋通り」が抽出されなかったのは，品詞分解すると「小滝」が普通名詞であり，5.1 節で示したグループ 1 に含まれないためである．「小滝橋通り」は表 1 に示したアドレスマッチングテーブルに含まれていないので，地名選択ルールを作成する際に影響していない．グループ 1 に普通名詞を加えれば抽出が可能になるが，適合率が大幅に低下してしまう．

適合率はどちらもほぼ 70%程度となった．主な誤りの原因は，「細麺」のような普通名詞や「山田」のような人名，「野間灯台」のような地名以外の固有名詞から始まる文字列を抽出してしまうためである．しかし，これらの地名以外の文字列は後処理としてアドレスマッチングを行えば，容易に選別することができる．

7. 空間データ抽出システム

アドレスマッチング手法と半構造化手法を組み合わせることにより，N-I データや S-I データから，S-D 空間データを作成することができる．つまり，自然言語で記述された情報から地図上の位置を抽出し，地図空間に射影することができる．この処理を確認するため，実験的な空間データ抽出システムを実装し，『芭蕉』と命名した．『芭蕉』は，入力情報が N-I データの場合は半構造化を行って S-I データに変換し，さらにアドレスマッチング手法により S-D データに変換する．入力情報が S-I データの場合はアドレスマッチング手法だけを適用して S-D データに変換する（**図 7**）．

7.1 『芭蕉』の実装

図7を用いて，『芭蕉』のモジュールとデータの流れを説明する．まず，4節のアドレスマッチング部をC++でUNIXワークステーション上にサーバプログラムとして実装した．起動するとデーモンプロセスとして待機し，アドレスマッチングを利用するクライアントアプリケーションがIP接続するのを待つ．クライアントアプリケーションから地名文字列が渡されると，地名参照テーブルから最長一致となる地名とその緯度経度を検索して返す．次に，5節の半構造化部をPerlでUNIXワークステーション上に実装した．さらにアドレスマッチング部と半構造化部を結合する部分を追加し，『芭蕉』とした．

『芭蕉』はN-Iデータが入力された場合，半構造化部を呼びだしてS-Iデータを作成する．S-Iデータを作成した後，またはS-Iデータがはじめから入力として与えられた場合，次の処理を行う．

(1) <spa>w</spa>で表現された部分を見つける．

(2) wをアドレスマッチングサーバに問い合わせて最長一致となる地名w_{best}とその緯度y，経度x，及び一致度を表す得点sを取得する．

(3) sが合格点s_t未満であれば以降の処理をスキップする(s_t = 3を既定値としているが，用途によって適合率を高くする場合はs_t = 4，再現率を高くする場合はs_t = 2とする)．

(4) <spa>w</spa>を<spa name="w_{best}" p="y x">w<author>basho ver2</author></spa>に置き換える．

『芭蕉』によって変換した例を**例2**に示す．

7.2 抽出されたデータの評価

『芭蕉』によりN-Iデータを S-Dデータに変換する際の精度について評価する．6.2で示した通り，N-Iデータからほぼ100%の再現率で地名を含む文字列を抽出しS-Iデータに変換することができる．しかし，そのうち30%程度は地名ではない文字列なので，アドレスマッチングを後処理として行うことで適合率を向上させる必要がある．そこで次の実験を行い，性能を評価した．

実験3. 実験2で利用したサンプル1の一部を『芭蕉』によってS-Dデータに変換し，再現率・適合率を人手により検査した．また，変換の際の閾値s_tを変化させて再現率と適合率の変化を調べた．この実験では，地名部分の文字列だけを抽出できた場合に適合度1，さらに緯度・経度が正しい場合に適合度2とする．

実験の結果を**表5**に示す．同じ文章を入力として閾値を変化させたところ，s_t = 1では25件の地名を抽出

	サンプル1	サンプル2
再現率	99.7	100.0
適合率	69.2	69.6

表4 半構造化手法による地名抽出率（%）
Table.4 Recall and precision rates (%) of geographic words extraction from plain text files

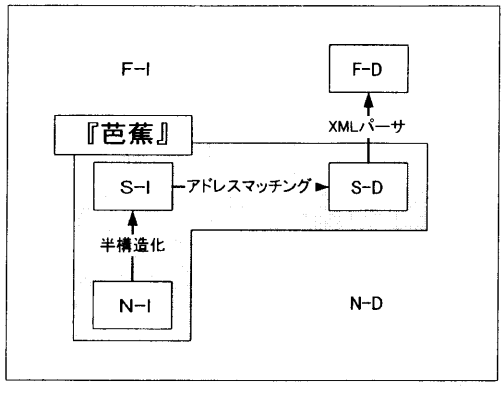

図7 データ変換システム『芭蕉』
Fig.7 Data conversion system "Basho"

<spa name="池ノ上駅">池ノ上駅</spa>前のパン屋のゴマ餡パンはお勧めです。

↓

<spa name="池ノ上駅" type="point" p="139.676636E 35.657230N">池ノ上駅<author>basho ver2 </author></spa>前のパン屋のゴマ餡パンはお勧めです。

例2 『芭蕉』による処理の例

	適合度1	適合度2
s_t = 1	25/58(=43.1%)	21/58(=36.2%)
2	32/48(=66.7%)	28/48(=58.3%)
3	19/23(=82.6%)	18/23(=78.3%)
4	8/8(=100.0%)	8/8(=100.0%)

表5『芭蕉』による地名抽出の適合率
Table.5 Precision rates of geographic words extraction using "Basho" from plain text files

できたが，$s_t = 4$ では8件の地名しか抽出できていない．しかし $s_t = 1$ の場合の適合率は，適合度1では48.1%，適合度2では36.2%であったものが，$s_t = 4$ では適合度1，適合度2ともに100.0%となっている．そこで，実際に応用する場合は，再現率が重要な用途では s_t を低く，適合率が重要な用途では s_t を高く設定する．

地名部分を‹spa›表現で指定してあるS-Iデータを入力として用いた場合，（記述ミスがない限り）適合度1は常に100%である．また，この場合の適合度2となる適合率は，地名をどの程度細かく記述しているかに依存する．例えば地名を住所で記述した場合は，該当する地名がアドレスマッチングテーブルに存在すれば100%の適合率が得られる．しかし，駅名など同名の地名が他にも存在する記述をした場合は，適合率が低下する．

7.3 『芭蕉』の適用例

『芭蕉』の適用例として，WWW上に存在する空間データを収集し，地図上にリンクするアプリケーションである，空間データサーチエンジンを開発した[9]（図8）．このアプリケーションは，データを収集するバックエンドと，データを検索するフロントエンドに分かれている．バックエンドでは，まず，WWWサーチロボットがWWWページ上のリンクを辿りながらページを収集する．集めたページを『芭蕉』の入力データとして与え，‹spa›表現のS-Dデータを得る．この出力結果をXMLパーサで処理し，空間座標値と元URLを要素とするテーブルを作成する．最終的に得られるテーブルは，点ベースの地理データとして利用できるF-Dデータである（例3）．

フロントエンドはCGIプログラムを利用した検索インタフェースであり，座標-URLテーブルを元に動的にクリッカブル地図画像を作成する．地図画像上のポイントはページがもともと存在したURLへのリンクになっているので，ポイントをクリックすると元のページにジャンプし，内容を表示することができる．図9〜11に利用画面を示す．

図9は小縮尺での利用例で，関東地方の温泉の分布を示したものである．元のページに市町村レベルまでの地名しか含まれていないため，位置の精度は数十km程度の曖昧さがあるが，温泉やスキー場のように空間的な広がりを持つものであればこのレベルでも十分に有用なデータとして利用できる．

図10は大縮尺の利用例として東京のラーメン屋の分布を示したもので，番地レベルでの位置精度を持っている．図10のポイントをクリックすると，図11のような詳細情報が表示される（図9, 11の下部の地図はMapFanWeb[16]に緯度経度情報を送り，地図画像を取得して表示している．また，図9, 10ともアドレスマッチングの閾値 $s_t = 3$ としている）．

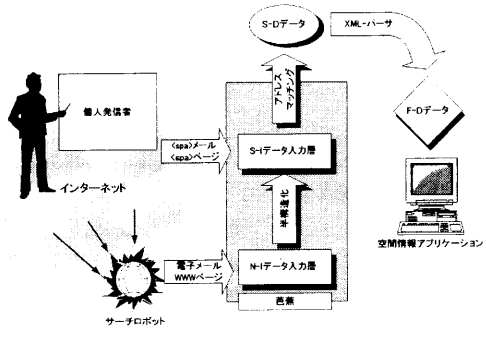

図8 『芭蕉』を利用した空間サーチエンジン
Fig.8 Spatial search engine using "Basho"

```
<spa name="池ノ上駅" type="point" p="139.676636E
35.657230N">池ノ上駅<author>basho ver2
</author></spa>前のパン屋のゴマ餡パンはお勧めです。
                    ↓
|x          |y          |URL
|-----------|-----------|--------------------------
|139.67     |35.66      |http://www.somewhere/a.html
```

例3 XMLパーサで処理して得られるF-Dデータ

8. まとめ

本稿で提案した手法はネットワーク上に限らず，過去の新聞記事データベースや文字放送など，自然言語で記述された空間的な位置を含む情報源から，F-Dデ

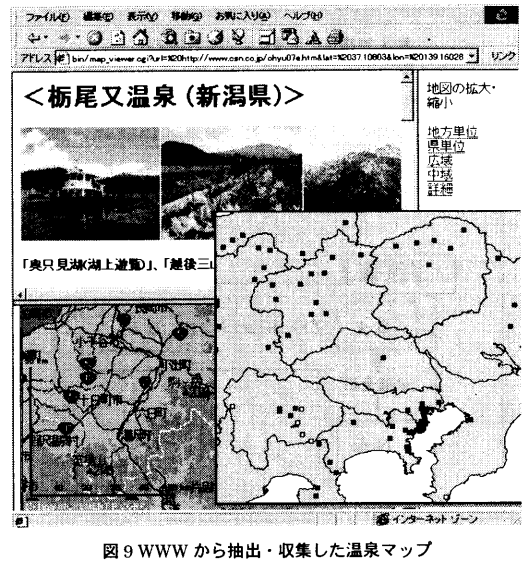

図9 WWWから抽出・収集した温泉マップ
Fig.9 Hot spring maps extracted from the Web

ータを作成するために利用することができる．また，‹spa›表現によって半構造化されていれば，動画像や音声といった理解・認識が困難な情報源からも地理データを作成でき，カーナビゲーションシステム等で需要が大きい「精度は低くても新鮮なデータ」を大量に収集することが可能になる．

また，インターネット上のWWWページや電子メールで空間情報を発信・流通するためのXML記述である，‹spa›表現を提案した．‹spa›表現は，個人レベルでも容易に利用できるシンプルな表現だが，ナビゲーションシステムなど多くの応用が考えられる．一般ユーザは空間的な位置を座標で記述することが困難なので，‹spa›表現では住所などの地理参照記述による位置の表現を認めている．日本全国をカバーするような広範囲なデータを扱うことが可能な二重管理構造を利用したアドレスマッチング手法を利用することで．地理参照で記述されていても，座標を含んだ記述に変換することができるため，座標で記述されたデータと同様にコンピュータで応用に利用できる．

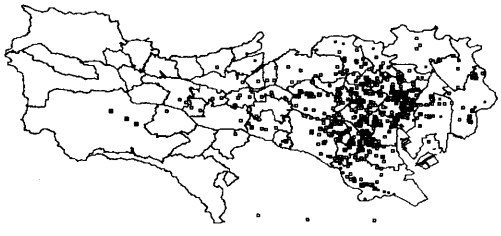

図10 WWWから抽出した東京のラーメン屋分布
Fig.10 Chinese noodle restaurants distribution extracted from the Web

図11 検索した情報例（ラーメン屋詳細）
Fig.11 Chinese noodle restaurants information page linked to maps

さらに，‹spa›表現を利用していない文書からでも，自然言語処理の技術を利用して含まれている地理参照情報を検索し，地理データとして利用する半構造化手法を示した．この半構造化手法とアドレスマッチング手法を組み合わせた空間情報媒介システム『芭蕉』を開発し，適用例として空間サーチエンジンを実装して有効性を確認した．また，アドレスマッチング手法の検索効率，半構造化手法の変換精度について実験を行い，十分実用的な性能を持つことを確認した．

謝辞 本研究のため，建設省国土地理院ならびに住友電工株式会社の地図データを利用させていただきました．また，本論文を改善する上で有益なご意見を頂いた，査読者の方々に感謝いたします．

参考文献

1) David J Maguire, Michael F Goodchild and David W Rhind, Geographical Information Systems, Longman Scientific & Technical, 1991
2) 三浦信幸・横路誠司・高橋克巳・島健一(1998)GISを用いた位置指向のWWWサーチエンジン～モーバイルインフォ2実験～，「地理情報システム学会講演論文集」，Vol.7，pp.131-136
3) W3C, Extensible Markup Language(XML), http://www.w3.org/XML/
4) Serge Abiteboul, Peter Buneman, Dan Suciu, Data on the Web, Morgan Kaufmann Publishers, 2000
5) Alin Deutsch, Mary Fernandez, Daniela Florescu, Alon Levy, Dan Suciu, XML-QL: A Query Language for XML, http://www.w3.org/TR/1998/-NOTE-xml-ql-19980819/, 1998
6) Open GIS Consortium,Inc.，“OpenGIS Simple Features Specification for SQL Revision 1.0”，http://www.opengis.org/public/sfr1/sfsql_rev_1_0.pdf
7) G-XML ホームページ，http://gisclh.dpc.or.jp/-gxml/contents/index.htm
8) Karen C.Hanna, R.Brian Culpepper, GIS in Site Design, John Wiley & Sons, Inc., 1998(Geo-reference, Address Matching)
9) 原田 豊，島田 貴仁，数値地図2500のための住所照合ユーティリティの開発，VCGIS'98，http://-queen5.mm.ics.saitama-u.ac.jp/~vcgis/vcgis/-abstract_j/harada-j.html, 1998
10) Environmental Systems Research Institute, ArcView GIS Users Guide
11) アドレスマッチングシステムデモページ，http://-www.csis.u-tokyo.ac.jp/~sagara/cgi-bin/dams.cgi
12) 松本裕治,北内啓,山下達雄,今一修,今村友明，“日本語形態素解析システム『茶筌』version 1.0使用説明書”，NAIST Technical Report, NAIST-IS-TR97007, February 1997, http://cactus.aist-nara-.ac.jp/lab/nlt/chasen.html
13) 東京のラーメン屋さん, http://www.torasan.com/
14) 全国温泉案内『Oh！YU』，http://www.csn.co.-jp/ohyuhm.htm#onsenpre）

15) 空間サーチエンジンデモページ, http://www.csis.u-tokyo.ac.jp/~sagara/cgi-bin/japan_map.cgi
16) INCREMENT P CORP(1996), MapFanWeb:-http://www.mapfan.com/

(平成12年 3 月20日受付)
(平成12年 6 月30日採録)

(担当編集委員 石川 博)

相良 毅(正会員)

1993 年埼玉大学工学部情報工学科卒業. 1995 年東京大学大学院工学系研究科情報工学専攻修士課程修了. 1998 年 6 月, 東京大学空間情報科学研究センター助手.

有川 正俊(正会員)

1986 年九州大学工学部情報工学科卒業. 1988 年同大学院工学研究科情報工学専攻修士課程修了. 同年九州大学大型計算機センター助手. 1989 年同大学工学部情報工学科助手. 1993 年京都大学工学部情報工学科助手. 1994 年広島市立大学情報科学部助教授. 1999 年東京大学空間情報科学研究センター助教授, 現在に至る.

坂内 正夫(正会員)

1975 年, 東京大学大学院工学系研究科博士課程修了. 同年, 同大学工学部電気工学科専任講師. その後, 横浜国立大学工学部情報工学科助教授, 東大生産技術研究所助教授を経て, 1988 年, 同大学生産技術研究所教授, 1994 年, 同大学概念情報処理工学研究センター長, 1998 年, 4 月より同大学生産技術研究所所長に就任.

専門, マルチメディア情報処理. 工学博士. 現在, 文部省創成的基礎研究「マルチメディア情報媒介機構の研究」プロジェクトリーダー, IEEE International Conference on Multimedia Computing and Systems, ITS Conference など, 7 つの国際会議の組織委員長, プログラム委員長等歴任.