TR-IT-0213

音声とポインティング・ジェスチャの統合意味解析

Integrated Analysis of Speech and Pointing Gesture

水梨 豪　　田中吾郎*

Suguru MIZUNASHI　　Goro TANAKA

1997.3

概要

本稿では、音声言語情報(聴覚のモダリティ)に加えて、視覚のモダリティであるポインティング・ジェスチャ、すなわち、地図上に指で描かれる点や線の情報を入力として利用できるマルチモーダル・インターフェースについて述べる。はじめに、入力された音声とポインティング・ジェスチャの統合的な意味を同定する枠組みについて述べ、その枠組を応用して試作したマルチモーダル問い合わせシステムについても簡単に言及する。

ATR 音声翻訳通信研究所

ATR Interpreting Telecommunications Research Laboratories

*株式会社CSK

目次

音声とポインティング・ジェスチャの統合意味解析

1　はじめに

音声翻訳通信、すなわち、異言語話者間で機械翻訳を介して行われる通信(対話)に関する研究においては、さまざまな克服すべき課題が存在する。それらの課題うち、現在もっとも重要と考えられているもののひとつに、「より自然な翻訳対話の実現」がある。音声翻訳通信システムを構成する要素技術である、音声認識技術、言語解析技術、機械翻訳技術、音声合成技術などのそれぞれの領域においても、日常人間同士の対話で発せられているような、音韻的にも文法的にもくだけた発話を認識・解析する手法、意味的に自然な翻訳結果を生成する手法、より人間らしい合成音声を生成する手法などが盛んに研究されている。

本研究も、より自然な音声翻訳対話の実現を最終的な目標としているが、本研究がその目的を達成するために用いるアプローチは、上記要素技術におけるそれとは性質が多少異なっている。すなわち、上記の各要素技術が、音声言語の認識、解析、翻訳、合成のそれぞれの領域においてより自然な発話を取り扱える枠組みを構築しようとしているのに対して、本研究では、音声言語をコミュニケーションの主軸に据えて、さらに音声言語以外の情報も同時に利用する(いわゆる「マルチモーダル情報」を利用する)ことによって、対話をより自然で効率的なものにするというアプローチをとっている。日常の人間同士の対話では、主に聴覚のモダリティ(声を聴く／発する)や視覚のモダリティ(身振り、手振り、表情などを見る／見せる)などの複数のモダリティが同時に使用されながら、自然な対話が成立している。近年、ユーザ・インターフェース研究などの領域では、計算機との対話手段としてこれらのような複数の情報を同時に利用して、人間と計算機の、ひいては計算機を介した人間同士の、より自然な対話を実現しようとする動きが盛んになってきている[1][2]。

本研究では、音声言語情報(聴覚のモダリティ)に加えて、視覚のモダリティであるポインティング・ジェスチャ、すなわち、地図上に指で描かれる点や線の情報を入力として利用できるインターフェースを想定している。地図などをはさんで、その上に指差しを行いながら対話するという行為は、日常頻繁に行われ、かつ、対話を円滑に進めていると考えられる。したがって、将来、異言語間の音声翻訳対話システムにおいてこのようなユーザ・インターフェースを導入することによって、より自然で効率的な音声翻訳対話環境の実現を期待することができる。

本稿では、そのようなマルチモーダル・ユーザ・インターフェースのいくつかの要素技術のうち、主に、入力された音声とポインティング・ジェスチャの統合的な意味を同定する枠組みについて述べる。さらに、その枠組を応用して試作した、マルチモーダル対話システムについても簡単に言及する。

2　研究の経緯

音声とポインティング・ジェスチャを受け付けるマルチモーダル・ユーザ・インターフェースを構築するためにこれまで行ってきた研究内容を概観する。

2.1　データ収集のための実験

マルチモーダル音声翻訳対話環境における人間のふるまいを調査するために、マルチモーダル対話シミュレータEMMI[3]を用いた対話環境のもとで、実験を2回行った[4]。使用したシミュレータは、日本人用、英米人用、通訳者用の端末からなり、日本人用、英米人用の端末では、音声、地図上のポインティング・ジェスチャ、テキストなど数種類の入力手段を用いて、翻訳を介して対話できるようになっている(図1参照)。実験は2回とも、日本人被験者が英米人被験者に対して道案内をするタスクのもとに行われた(図2参照)。最初の実験は、人間の通訳者を介した対話実験であり、通訳者用の端末

に通訳者を配置し、9対話の収集を行った[5]。2番目の実験は、機械翻訳を模擬した対話実験で、最初の実験と同様に、通訳者用の端末には通訳者を配置したが、機械翻訳環境を模擬するために、Wizard of Oz方式(通訳は機械がしていると被験者に信じさせて行う実験方法)を採用し、通訳者は機械を装った反応や発話を行った。この実験では10対話の収集を行った[6]。

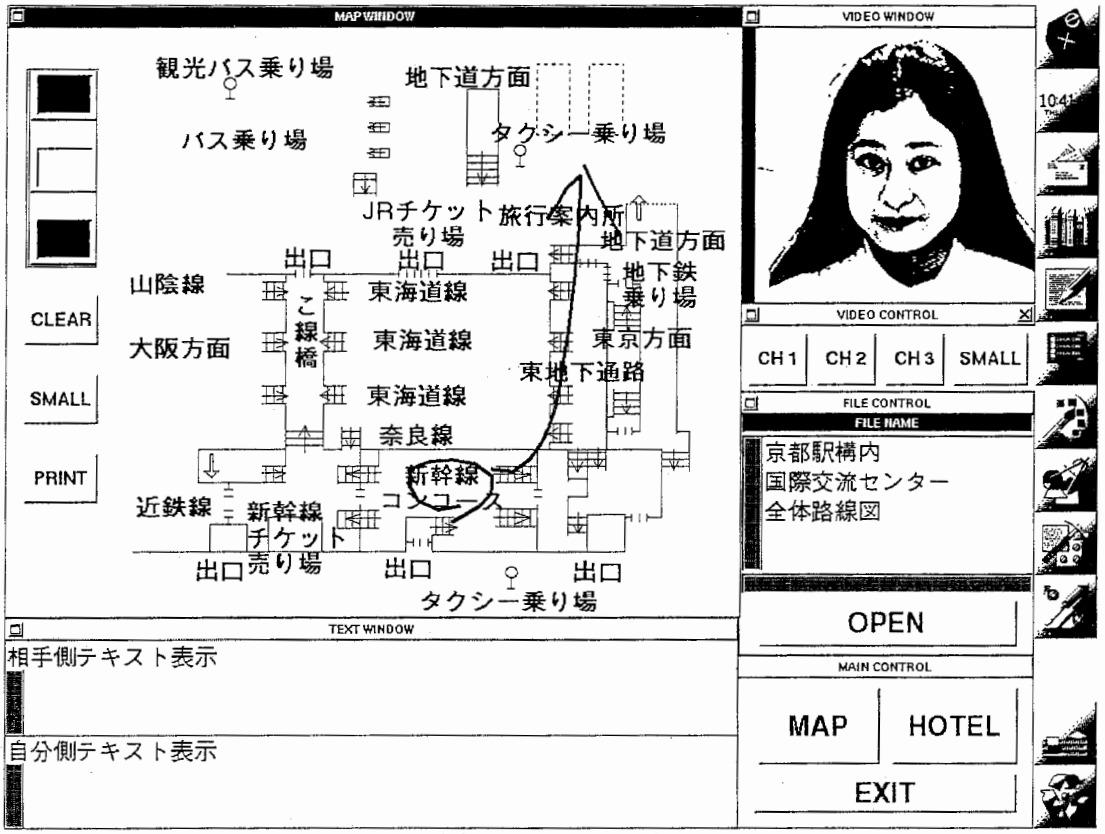


図1　EMMIの画面

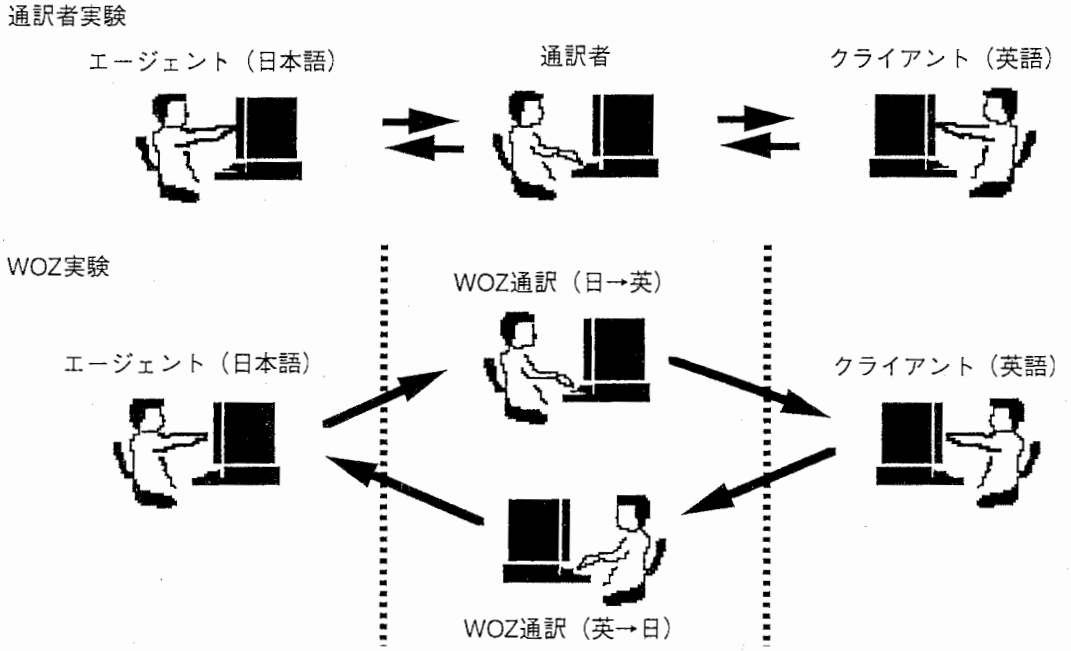


図2　ふたつの実験方法

2.2　データ分析

実験で収集した発話とポインティング・ジェスチャのデータを、以下の項目で分析した。

2.2.1　2回の実験データの差異

人間の通訳者を介した場合と機械翻訳を(模擬的にだが)介した場合の差異を分析した[4]結果、まず、前者よりも後者の発話量が少ないことが判明した。同等のゴール(ある場所までの道案内を受ける)を設定しているにもかかわらず、発話量が減少するのは、機械翻訳対話の効率の良さを示していると考えられる。次にわかったことは、ポインティング・ジェスチャの使用率をみると、後者は前者の3倍近くにのぼるということであった。これは、機械翻訳環境ではポインティング・ジェスチャはコミュニケーションの有力な手段となりうることを示していると考えられる。以上より、マルチモーダル機械翻訳対話においては、ジェスチャを用いた効率的な対話が可能であるという見通しが得られた。

2.2.2　発話とポインティング・ジェスチャの関連性の分析

収集したデータのうち、ジェスチャをともなう発話を抽出し、発話とポインティング・ジェスチャの関連性を分析した[4]。これにより、以下のような、発話とジェスチャを統合的に解析する枠組みを構築するための基本的な資料となる事項が明らかになった。

(1)地図を用いた道案内タスクにおいて主に使用されるジェスチャは、オブジェクトを丸で囲む方法(サークリング・ジェスチャ)や道などに沿って線を引く方法(ラインドラッギング・ジェスチャ)などの5種類であり、各ジェスチャは位置の提示、経路の提示など、固有の意味を持っていることがわかった。

表1　ジェスチャの種別と指示内容

種別	形状	指示内容
サークリング		もの
ポインティング		
マーキング		位置
スクランブリング		
ラインドラッギング		経路

(2)各ジェスチャは、ある特定の語句(主に指示語を含む語句)とともに使用され、その語句(以下、ジェスチャ対応語句と呼ぶ)に対して指示内容を意味的に補填する形で対応し、最終的に発話とジェスチャ両者の統合的な意味が同定されることがわかった。例として、「このホテルがいちばん安いです」という発話とともに地図上の都ホテルを丸で囲む場合を考える。最終的な「都ホテルがいちばん安い」という意味は、そのサークリングジェスチャが都ホテルという「もの」を指示しており、さらにその指示内容は「このホテル」に対して補填されるということがわかって初めて同定される。「このように行ってください」という発話と、経路を表わすラインドラッギングジェスチャの組み合わせの場合も同様で、「このように」という語の意味内容として、「ジェスチャの線で表わされる具体的な経路」という指示内容が補填される。以上からわかるように、発話とジェスチャの統合的な意味を同定するためには、

1.ジェスチャの指示内容の同定

2.ジェスチャが対応する発話中の語句の同定

という作業が重要である。

(3)ジェスチャの役割は、大別して、「発話に欠如している情報を補完すること」と「発話と重複した情報を提供することによって発話+ジェスチャの全体の意味を確実にすること」であることがわかった。前者の例としては、「京都ホテルはこちらです」と発話しながら地図上の京都ホテルを丸で囲んだり、「このように行ってください」と発話しながら経路を線で描く事例がある。この場合、

「こちら」や「このように」という語の意味は、ジェスチャの情報なしでは理解できないので、ジェスチャの情報を用いて意味を補完する必要がある。後者の例としては、「京都ホテルにはまだお部屋があります」と発話しながら京都ホテルを丸で囲んだり、「東地下通路を通ってください」と発話しながら東地下通路上に線を描く事例がある。この場合は、発話の情報だけから意味が通じるので、ジェスチャを用いて「京都ホテル」や「東地下通路」を示すことは、一見すると、発話＋ジェスチャ全体からみれば情報が重複していることになり、ジェスチャの情報は確認の意味しか持たないように見える。しかしながら、別の見方をすると、二通りの情報をもとにいずれかの情報の曖昧性(音声または言語的曖昧性とジェスチャの曖昧性)を解消できるという可能性もあることがわかった。

3　音声とジェスチャの意味統合システム

3.1　システム概要

分析結果をふまえて、音声とジェスチャの統合的な意味を同定するシステムを作成した。図3にこのシステム(SparcStation 10上のX Windowに実装されている)のグラフィカル・ユーザ・インターフェースを示す。ユーザは、マイクを通して音声を入力できるとともに、タッチパネルを通して地図上に指で線を描く(ジェスチャを行う)ことができる。システムは最終的に、入力された音声とジェスチャの統合的な意味を同定する。例えば、ユーザが「このホテルに部屋はありますか」という発話をするとともに、地図上の京都ホテルを丸で囲んだ場合、システムは発話とジェスチャの時間的な関係をグラフで表示し(画面左上)、両者の統合的な意味の同定結果を出力する。

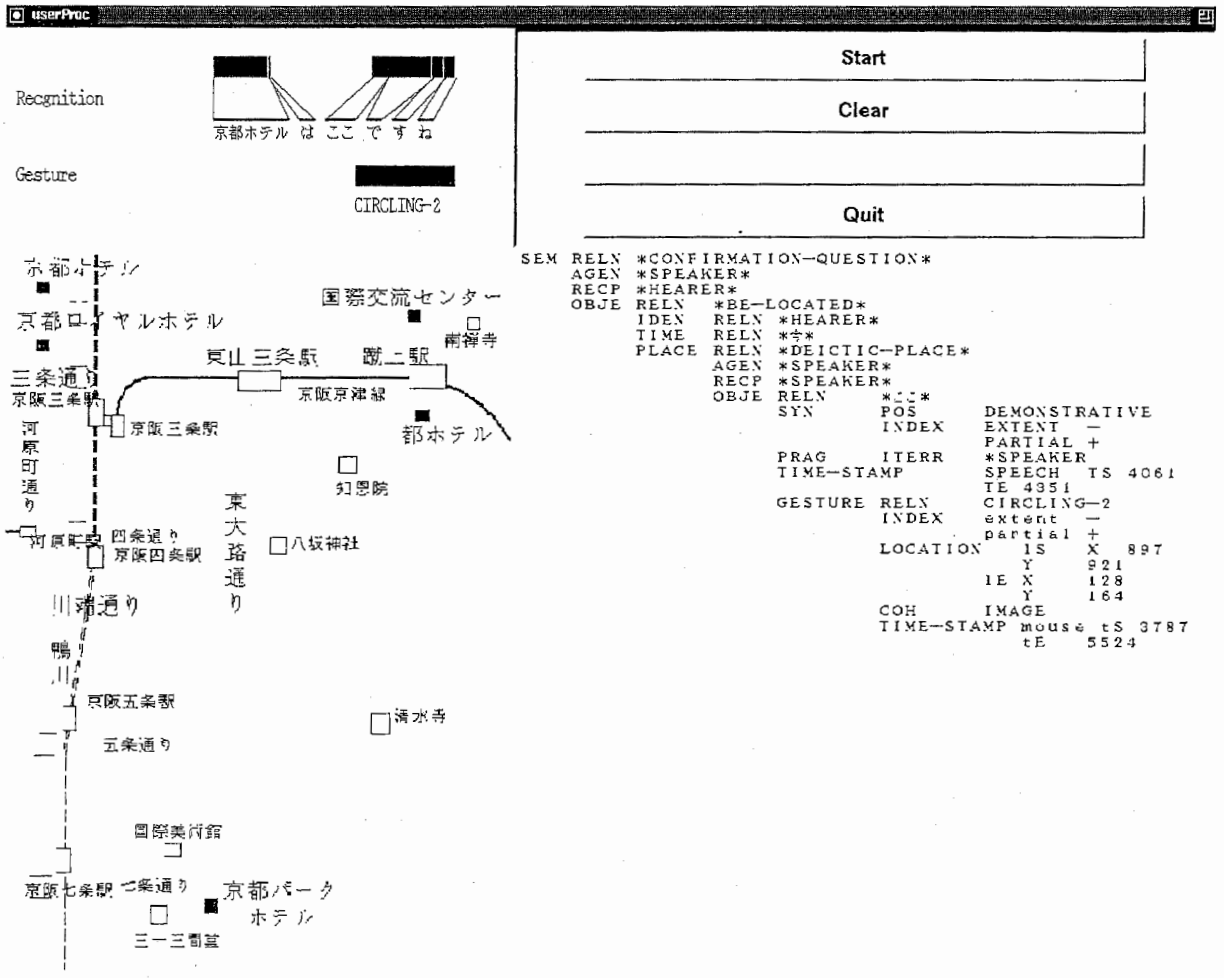


図3　システムの画面

システムの構成を図4に示す。スーパーバイザが他のすべてのモジュールを管理している。マイクを通して入力されたユーザの発話は、音声認識部と自然言語解析部を通って、意味表現に変換される。タッチパネルを通して地図上に描かれたジェスチャは、ジェスチャ解析部で解析され、形状や指示対象候補が同定される。その後、発話とジェスチャ両者の意味が統合意味同定部で同定される。

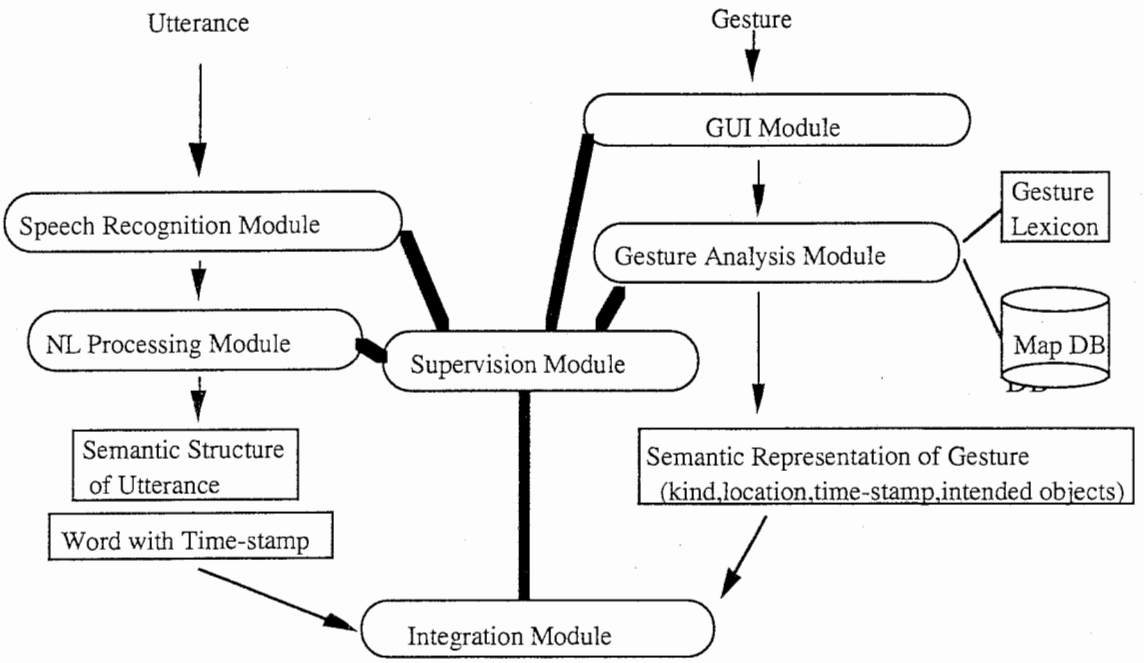


図4　システム構成

以降に各モジュールの詳細を述べる。

3.2　スーパーバイザ

本システムの動作をコントロールするモジュールである。具体的には、以下の機能を持つ。

(1)ユーザ・インターフェースのPEDALボタンのonと同時に音声認識プログラムに音声の取り込みを指示する。

(2)ジェスチャが入力された場合、そのジェスチャデータをジェスチャ解析部に渡す。

(3)PEDAL ボタンの off を検知して、音声認識部からの日本語文字列の出力を待つ。

(4)音声認識部の結果出力を検知して、データを日本語解析部に渡し、その出力を待つ。

(5)日本語解析部の出力とジェスチャ解析部の出力が両方ともなされたことを検知して、それらのデータを統合処理部に渡し、その出力を待つ。

3.3　グラフィカル・ユーザ・インターフェース

グラフィカル・ユーザ・インターフェース・モジュールの機能を以下に挙げる。

(1)ジェスチャが行われる対象となる、イメージを表示する(今回は京都市街の地図)。

(2)タッチパネルによって、ジェスチャの入力、ボタンの押下を受け付ける。

(3)PEDAL ボタンの状態を検知することによって、処理開始/処理終了をシステムに通知する。

(4)入力されたジェスチャの種類(後述の5種類)を表示する。

(5)システムの処理状況を表示する。

(6)CLEARボタン押下を検知して、描かれたジェスチャ軌跡やジェスチャ種類表示をクリアする。

(7)QUITボタン押下を検知して、システムを終了する。

3.4　発話解析部

3.4.1　音声認識部

ユーザはマイクに音声を入力する。マイクはA/D変換器(DATLINK)に接続されており、音声信号はディジタル信号に変換される。その後、音声データはまずnarecordというプログラムで処理される。このプログラムは、DATLINKから音声データを取り出すプログラムである。取り出されたデータはspteeというプログラムに渡され、音声入力の切り出しを行なう。narecordからの音声データ入力は常時おこなわれているが、このspteeプログラムはユーザ・インターフェースのPEDALボタンのon/off状況に合わせてnarecordのデータをon/offするゲートの役割を持っている。つまり、PEDALボタンがonの場合のみ音声データを次の処理を行なうプログラムに渡すのである。処理対象となったデータは、ATR音声翻訳通信研究所第1研究室が開発したSPREC[7]という音声認識プログラムで処理され、最終的に音声認識結果(日本語文字列)と単語ごとのタイムスタンプが出力される。

3.4.2　言語解析部

言語解析部では、形態素解析、構文解析、依存構造解析、格構造解析、意味解析を順番に行っている。それぞれの解析ツールは、ATR音声翻訳通信研究所第4研究室で開発されたものである[8]。今回は、入力想定文(付録3)にあわせて、形態素解析用の辞書、構文解析用の文脈自由文法、依存構造解析用の注釈付き文脈自由文法、格構造解析と意味解析用の構造書換ルールを作成して使用した。

図5に、「ここで左の曲がって川端通りを真っ直ぐ進んでください」という文が入力されたときの、意味解析部からの出力の例を示す。出力される意味表現の中で使用する各ラベルは、ほぼ[9]に準拠している。

```
[SEM [[RELN 並列表現]
[ARG1 [[RELN *ACTION-REQUEST*]
        [AGEN *SPEAKER*]
        [RECP *HEARER*]
        [OBJE [[RELN *曲がる*]
                [SUBJ *HEARER*]
                [LOCT [[RELN *ここ*]]]
                [OBJE [[RELN *左*]]]]]]]
[ARG2 [[RELN *ACTION-REQUEST*]
        [AGEN *SPEAKER*]
        [RECP *HEARER*]
        [OBJE [[RELN *進む*]
                [SUBJ *HEARER*]
                [ROUT [[RELN *川端通り*]]]
                [MANN [[RELN *真っ直ぐ*]]]]]]]]
```

図5　言語解析部の出力例

3.5　地図データベース

3.5.1　オブジェクトの種類

地図データベースには、地図上の建物、道路など(以降、オブジェクトと呼ぶ)の情報が登録されている。オブジェクトは、その性質によって次の3種類に分類されている。

・点的オブジェクト

　建物、駅などのように、広さの情報を持たないもの。

・線的オブジェクト

道路、鉄道などのように、長手方向の位置情報が参照される可能性があるもの。

・面的オブジェクト

公園などのある程度広い敷地、行政区画、商店街などのように、広さを持ち、その広さゆえにそのオブジェクトの内部の情報が参照される可能性があるもの。

実際、あるオブジェクトが点的なのか線的なのか面的なのかは、ユーザに見えている大きさ、つまり、地図の縮尺に依存する。今回はその判断については行わず、現状で使用している地図の縮尺で考え、オブジェクトを分類している。

3.5.2 オブジェクトが保持する情報

地図上の各オブジェクトは以下の情報を持つ。

・ID番号

・名称(左京区,蹴上駅,清水寺,京都ホテル,三条通り…)

・地図上の位置(矩型のふたつの頂点で代表させている)

・種別(地域,駅,寺,ホテル,道路…)

オブジェクトがホテルの場合は、空室情報と価格情報も付加してある。

3.6 ジェスチャ解析部

タッチパネルを通して入力されたジェスチャは、地図上の点の集合として保存され、それをもとにジェスチャの形状の認識と指示内容の同定を行っている。

3.6.1 形状認識

入力されたジェスチャは、以下の手順で「サークリング」「ラインドラッギング」「マーキング」「ポインティング」「スクランブリング」の5種類に分類される。

(1) ジェスチャを構成する点列のうち、重複するものを排除する。

(2) 点を全て含む最小矩形領域を定義し、その矩形に対する点の密度が90%以上である場合は「ポインティング」と判定する。また、点の数が5点以下で密度10%以上のものに対しても「ポインティング」と判定する。

(3) 以下のすべての条件に合致したものを「マーキング」と判定する。

a. 点列を結ぶ隣り合う線分同士がなす角度をθとし、$\cos\theta$の極値がただ1つ(折点が唯一)である。

b. 始終点間の水平距離が矩形の水平方向の辺の長さの70%以上である。

c. 垂直方向で折点の存在する側と反対側25%の部分に始終点双方が存在する。

(4) この時点で種類が確定していないジェスチャに対して、ジェスチャ構成点が3点以下のものは「ラインドラッギング」と判定する。

(5) ジェスチャ構成点の間を線分で結び、その線分上の点列を構成点として補間・追加する。

(6) 補間の完了したジェスチャ図形上の点から2の矩形領域中心点に向かって引かれる直線がジェスチャ図形を横切る回数を求める。

(7) 全ての直線がジェスチャ図形と3点以上で交わっている場合、「スクランブリング」と判定する。

(8) 図形と2点で交わった直線が全体の87.5%以上あった場合、「サークリング」と判定する。また、図形と1点でしか交わらない直線が全体の15%以下かつ2点で交わる直線が全体の75%以上の場合も「サークリング」と判定する。また、1点でしか交わらない直線がなく、2点で交わるものが40%以上の場合も「サークリング」と判定する。

(9) 1点でしか交わらないものが全体の70%を超える場合、「ラインドラッギング」と判定する。

(10) この時点で種類が確定していないジェスチャ図形に対して、水平直線、垂直直線を引いたときそれらの線が図形を横切る回数を求める。

(11) 1点でしか交わらないものが水平/垂直線全体の70%以上であるとき「ドラッギンング」と判定

する。

(12) 3点で交わったものがなく、1点で交わったものが全体の30%以下であるとき「サークリング」と判定する。

(13) 以上の手順によって判定されなかったものは全て「ラインドラッギング」と判定する。

3.6.2 指示内容解析

3.6.2.1 ジェスチャの指示内容の表現

2.2.2で述べたように、ジェスチャの指示内容としては、「オブジェクト」、「位置」、「経路」がある。ここでは、それぞれのジェスチャの指示内容を次のように表現する。

1)「オブジェクト」の指示

[MOBJ [[ID n1][ID n2]…]]

主にライン・ドラッギング・ジェスチャ以外のジェスチャが指示する内容である。ここで、n1などは地図データベース中のオブジェクトのID番号を表わす。

2)「位置」の指示

[MPART [[ID n][MAREA [[X1 x1][Y1 y1][X2 x2][Y2 y2]]]]

サークリング・ジェスチャが指示する内容である。ここで、(x1,y1)と(x2,y2)はそれぞれ、サークリングジェスチャの外接矩型の対角点を表わし、全体として、「IDがnのオブジェクトのうち、サークリング・ジェスチャで囲まれた領域」を示す。

3)「経路」の指示

```
[[START [MOBJ [ID n1]]]                    # 経路の起点
 [MOVE [[FROM [MOBJ [ID n2]]]
        [TO [MOBJ [ID n3]]]
        [ALONG [MOBJ [ID n4]]]             # 経路
        [PASS [MOBJ [ID n5]]]]             # 経路上のオブジェクト
 [TURN [[AT [MOBJ [ID n3]]]                # 進路の転回点
        [DIRECTION 270]]                   # 進路の転換方向(角度)
 [MOVE [[FROM [MOBJ [ID n3]]]
        [TO [MOBJ [ID n6]]]
        [ALONG [MOBJ [ID n7]]]]
 [END [MOBJ [ID n6]]]]                     # 経路の終点
```

ライン・ドラッギング・ジェスチャが指示する内容である。ライン・ドラッギング・ジェスチャは、他のジェスチャがオブジェクトやその位置を指示するのに対して、経路を指示する。ここで言う経路とは、「AからBに行って、左に曲がって、Cまで行く」という移動の過程である。このような過程を表現するために、上述のような移動(MOVE)、転回(TURN)などのラベルを用いた。

3.6.2.2 指示内容の同定

ジェスチャの形状を認識した後、そのジェスチャが指示している内容を同定する。そのジェスチャがサークリング・ジェスチャならば、描かれたその丸の内部に位置するオブジェクトが指示されていると解釈するのが自然であるし、ライン・ドラッギング・ジェスチャならば、描かれたその線に沿っている地図上の道路などの上を移動するということが指示されていると考えれられる。しかしながら、ユーザがジェスチャを行う際は、指示したいオブジェクトを正確に丸で囲んだり、道路の上を正確になぞったりするとは限らない。そこで、指示内容を解析する場合は、ユーザのジェスチャにはある程度曖昧性があるという仮定のもとに、ひとつのジェスチャに対して複数の指示内容の候補を抽出することにした。例えば、サークリング・ジェスチャならば、丸の中に入っているオブジェクトとともに、丸が横切っているオブジェクトも指示されている可能性もあると考えて候補に入れておく。以下に、各ジェスチャの指示内容の同定方法を詳術する。なお、各ジェスチャは、ジェスチャが行われるオブ

ジェクトの性質によって指示内容が異なるので、指示内容同定の際には、3.5.1で分類した3種類のオブジェクトについて別々に記述する。

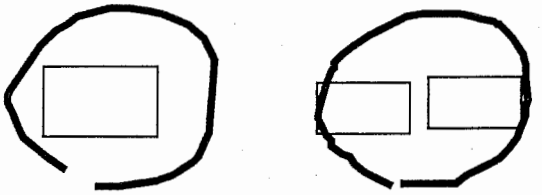
3.6.2.2.1　サークリング・ジェスチャ

(1)点的オブジェクト(建物、駅など)

1.ジェスチャの外接矩型のなかに含まれているオブジェクトを抽出する。
2.自身にサークルを含んでいるオブジェクトを抽出する。
3.サークルの中心とオブジェクトの中心の距離が小さい順番でソートする。
4.最短距離のオブジェクト(複数可)を第一集団として[MOBJ1 [[ID n1][ID n2]…]]のようにまとめる。
5.それ以外のオブジェクトを同様に[MOBJ2 [[ID m1][ID m2]…]]とまとめる。

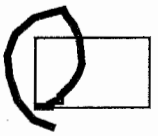
ジェスチャの使用例

大半の部分が囲まれている



[MOBJ [[ID n1][ID n2]…]]

一部が囲まれている



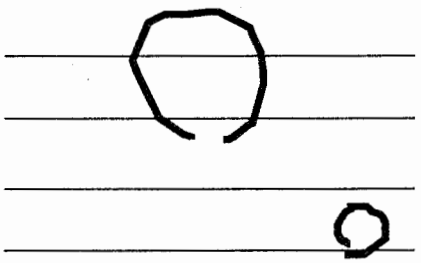
[MOBJ [[ID n]]]

オブジェクト内部にサークルがある

[MOBJ [[ID n]]]

(2)線的オブジェクト(道路など)

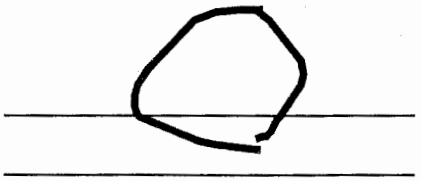
1.ジェスチャが触れている線的オブジェクトを抽出する。
2.サークルの中心と線的オブジェクトの中心線との距離の最小のオブジェクトを抽出する。
3.サークルの中心が道幅内にある場合、
[MOBJ [[ID n]]]
[MPART [[ID n][MAREA [[X1 x1][Y1 y1][X2 x2][Y2 y2]]]]]
のふたつの指示内容候補を生成する。
使用例



4.サークルの中心が道幅内にない場合、

[MOBJ [[ID n]]]

を生成する。



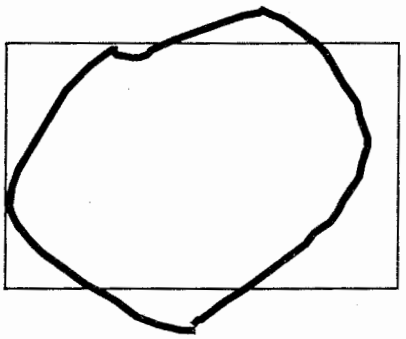
(3)面的オブジェクト(公園などのある程度広い敷地、行政区画など)

1.ジェスチャ矩型の大きさが自身の矩型の大きさの80%以上で、かつ、矩型の中心同士が近いオブジェクトを抽出し、

[MOBJ [[ID n1][ID n2]…]]

を生成する。

使用例



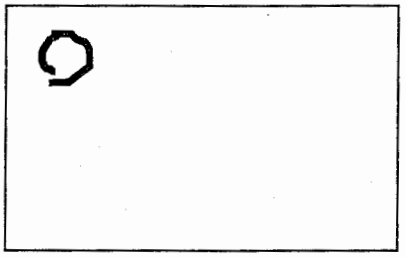
2.それ以外で触れているオブジェクトを抽出し、

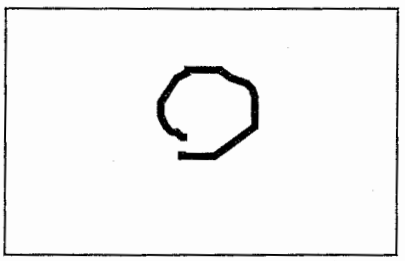
[MOBJ [[ID n]]]

[MPART [[ID n][MAREA [[X1 x1][Y1 y1][X2 x2][Y2 y2]]]]]

を生成する。

使用例





(4)サークリング・ジェスチャの指示内容の解析例

図6のような地図上になされたサークリング・ジェスチャの指示内容を解析すると、図7のような解析結果が得られる。一般に、ひとつのサークリングジェスチャがなされると、地図上のオブジェクトとの関連をもとに指示内容の候補が複数生成される。

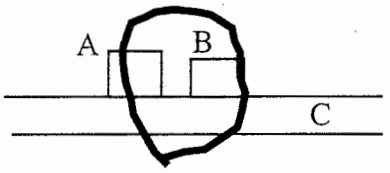


図6　サークリング・ジェスチャの例

[MOBJ [[ID nA][ID nB]]]
[MOBJ [[ID nC]]]
[MOBJ [[ID n中京区]]]
[MOBJ [[ID n地図]]]
[MPART [[ID n中京区][MAREA [[X1 x1][Y1 y1][X2 x2][Y2 y2]]]]]
[MPART [[ID n地図][MAREA [[X1 x1][Y1 y1][X2 x2][Y2 y2]]]]]

図7　解析結果

3.6.2.2.2　　ライン・ドラッギング・ジェスチャ

(1)ジェスチャの分割

3.6.2.1のような表現を得るために、以下の手順でジェスチャを転回点で分割する。

1. ジェスチャの始点(s)からジェスチャ構成点(pn)に引いた直線とその間の各点(s～pn)との距離を計算する。
2. 計算した距離の平均が 10.0 以上の場合、(s)から(pn)の区間は直線ではないと判定する。
3. 直線であると判定された場合、(pn+1)点を 1 の(pn)点として計算を繰り返す。
4. 直線でない場合、(s)～(pn)までの点のうち 1 で引いた直線と一番離れている点を転回点(pi)とする。
5. 直線(s)(pn)のどちら側に折点(pi)が存在するか計算し、ジェスチャが左右どちらに曲がっているか判定する。
6. (s)～(pi)をひとつの分割区間とする。
7. (pi)を新たに(s)として 1 からの処理を繰り返し、最終点(e)まで処理を行なうことによって、ラインドラッギングジェスチャが転回点と区間の集合という構造に分割される。

(2)分割区間ごとの解析

1. 全体の始点と終点を求め、STARTとENDとする。
2. 分割区間ごとに、始点、終点、通過点にあるオブジェクトと、線が沿っている道路を抽出し、それぞれ、MOVEの引数中のFROM、TO、PASS、ALONGの引数とする。
3. 転回点は、TURNの引数中のAT(転回点)とDIRECTION(転回方向)によって表現する。

図7と図8にそれぞれライン・ドラッギング・ジェスチャの例とその解析結果を示す。

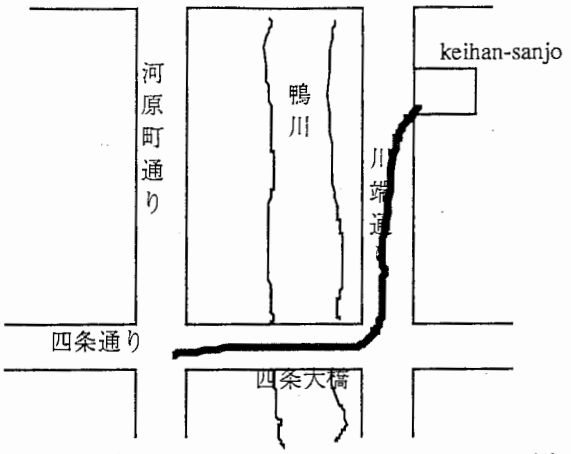


図7　ライン・ドラッギング・ジェスチャの例

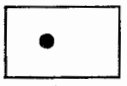
[[START [MOBJ n四条河原町]]
[MOVE [[FROM [MOBJ n四条河原町]]
　　[TO [MOBJ n四条川端]]
　　[ALONG [MOBJ n四条通り]]
　　[PASS [MOBJ n四条大橋]]]
[TURN [[AT [MOBJ n四条川端]]
　　[DIRECTION 270]]# 値を角度で示す
[MOVE [[FROM [MOBJ n四条川端]]
　　[TO [MOBJ n京阪三条]]
　　[ALONG [MOBJ n川端通り]]
　　[PASS NULL]]
[END [MOBJ n京阪三条]]]

図8　解析結果

3.6.2.2.3　　ポインティング・ジェスチャ

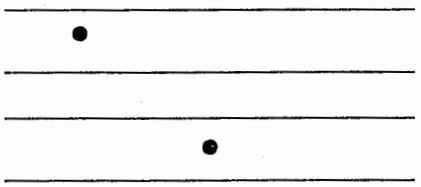
(1)点的オブジェクト

　ポイントが含まれているオブジェクトを抽出し、
　[MOBJ [[ID n]]]
を生成する。
　使用例



(2)線的オブジェクト

　ポイントが含まれているオブジェクトを抽出し、
　[MOBJ [[ID n]]]
　[MPART [[ID n][MPOINT [[X x][Y y]]]]]
を生成する。
　使用例



(3)面的オブジェクト

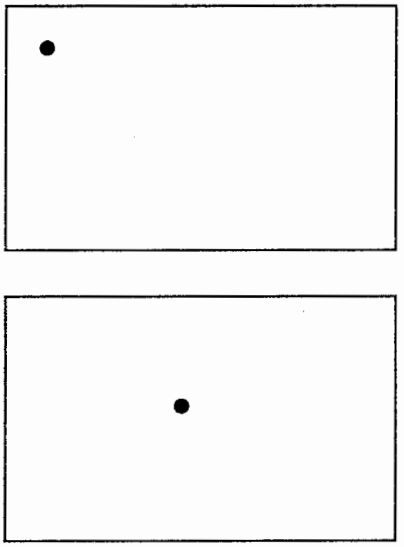
ポイントが含まれているオブジェクトを抽出し、

[MOBJ [[ID n1]]]

[MPART [[ID n][MPOINT [[X x][Y y]]]]]

を生成する。

使用例



3.6.2.2.4　スクランブリング・ジェスチャ

ジェスチャ座標の重心を求め、それをポイントと解釈して3.6.2.2.3と同様に解析する。

3.6.2.2.5　マーキング・ジェスチャ

ジェスチャ座標の重心を求め、それをポイントと解釈して3.6.2.2.3と同様に解析する。

3.7　統合意味同定部

統合意味同定部は、発話解析部から出力された発話の意味表現と、ジェスチャ解析部から出力された指示内容をもとに、両者の対応関係を同定する。ジェスチャは、ある特定の語句(主に指示語を含む語句)とともに使用され、その語句(以下、ジェスチャ対応語句と呼ぶ)に対して指示内容を意味的に補填する形で対応し、この結果、発話とジェスチャ両者の統合的な意味が同定される。例として、「このホテルがいちばん安いです」という発話とともに地図上のXホテルを丸で囲む場合を考える。最終的な「Xホテルがいちばん安い」という意味は、そのサークリング・ジェスチャがXホテルという「もの」を指示しており、さらにその指示内容は「このホテル」に対して補填されるということがわかって初めて同定される。「このように行ってください」という発話と、経路を表わすラインドラッギング・ジェスチャの組み合わせの場合も同様で、「このように」という語の意味内容として、「ジェスチャの線で表わされる具体的な経路」という指示内容が補填される。本節では、以上のようなジェスチャと発話の対応関係の同定の方法を、各ジェスチャごとに述べる。

3.7.1　サークリング・ジェスチャと発話の対応関係の同定

3.7.1.1　ジェスチャ対応語句の候補の抽出

まず、発話の意味表現から、ジェスチャ対応語句になりうる語句、すなわち、命題内容の述語の引数中の、指示語を含む語句と固有/普通名詞を抽出する。命題内容の述語の引数としては、以下の素性のみを考える。なお、素性が並列構造を持っているときは、並列するそれぞれをひとつの値として抽出する。

BE-LOCATED/OBJE
BE-LOCATED/LOCT
COPULA/IDEN
COPULA/OBJE
上記以外の述語/LOCT
上記以外の述語/DEPT
上記以外の述語/DEST

例えば、「京都ホテルはこちらにあります」という発話の、

```
[SEM [[RELN *INFORM*]
      [AGEN *SPEAKER*]
      [RECP *HEARER*]
      [OBJE [[RELN *BE-LOCATED*]
            [OBJE [[RELN *京都ホテル*]]
            [LOCT [[RELN *こちら*]]]]]
```

という意味表現からは、

```
*BE-LOCATED* :
        OBJE : [[RELN *京都ホテル*]]
        LOCT : [[RELN *こちら*]]
```

が抽出される。一般に、対応語句の候補は、

```
述語1:
        語句11
        語句12
        …
述語2:
        語句21
        語句22
        …
…
```

のかたちで抽出される。

3.7.1.2　述語に対応するサークリング・ジェスチャの同定

ひとつの発話の意味表現は、内部に複数の述語を持ちうる。例えば、「このホテルは高いですが、こちらのホテルは安いです」という発話の場合、「このホテル」をOBJE素性として持つ「高い」という述語と、「こちらのホテル」をOBJE素性として持つ「安い」という述語が並列で連接している。

一方、サークリング・ジェスチャに関しても、ひとつの発話中で複数回行われる可能性がある。したがって、ジェスチャに対応する語句を同定する前に、そのジェスチャと対応する述語を選定しておく必要がある。その作業を、3.7.1.1で抽出した語句のそれぞれについて、3.6.2.2.1で述べたようなサークリング・ジェスチャの各指示内容候補との関連尤度を計算することによって行う。具体的には、各語句ごとについて次のような手順で関連尤度を計算する。

1. 語句が発話された時刻の前後1秒以内に行われたジェスチャとの対応尤度に1を加算する。それ以外は0とする。
2. 語句が指示詞を含まない(固有名詞または普通名詞の)名詞句の場合、その語句の種別がMOBJの種別と一致するサークリングジェスチャとその語句との対応尤度に1を加算する。それ以外は0とする。
3. 語句が指示詞を含む語句の場合、その語句と各サークリングジェスチャとの対応尤度に1を加算する。さらに、その語句中に固有名詞または普通名詞が存在し、かつ、その種別がMOBJの種別と一致するサークリングジェスチャとその語句との対応尤度に1を加算する。さらに、
   その語句がBE-LOCATED/OBJならば同一述語単位内の*BE-LOCATED*/LOC
   その語句がBE-LOCATED/LOCならば同一述語単位内の*BE-LOCATED*/OBJ
   その語句がCOPULA/IDENならば同一述語単位内の*COPULA*/OBJ
   その語句がCOPULA/OBJならば同一述語単位内の*COPULA*/IDEN
   の種別がMOBJの種別と一致するサークリングジェスチャとの対応尤度に1を加算する。

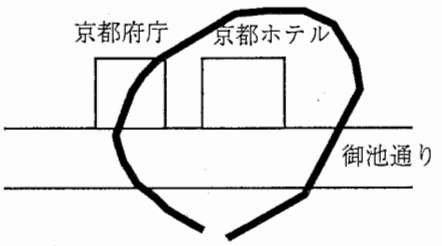
以上の手順をすべてのジェスチャ対応語句の候補について行い、各述語単位ごとに各サークリングジェスチャとの対応尤度を加算によって算出し、最尤なものを各述語単位の対応サークリングジェスチャとする。

3.7.1.3　ジェスチャに対応する語句の同定

対応サークリング・ジェスチャに対する対応尤度の最も大きな語句をサークリングジェスチャ対応語句とする。サークリング・ジェスチャの指示内容の候補が図9のように表現されているとすると、

[[RELN *京都ホテル*]]とサークリングジェスチャの対応尤度は1
[[RELN *こちら*]]とサークリングジェスチャの対応尤度は2

となるので、尤度の高い[[RELN *こちら*]]がジェスチャ対応語句と判定される。



[MOBJ [[ID n京都ホテル]]]
[MOBJ [[ID n京都府庁]]]
[MOBJ [[ID n御池通り]]]
[MOBJ [[ID n中京区]]]
[MOBJ [[ID n地図]]]
[MPART [[ID n中京区][MAREA [[X1 x1][Y1 y1][X2 x2][Y2 y2]]]]]
[MPART [[ID n地図][MAREA [[X1 x1'][Y1 y1'][X2 x2'][Y2 y2']]]]]

図9　サークリング・ジェスチャの例と指示内容候補

3.7.1.4　ジェスチャ指示内容の決定

前項までで、あるジェスチャに対応する語句が同定されたので、最後に、その語句と最終的に対応するジェスチャの指示内容を候補から選び出す。選び出す基準は、ある語句に対応すべきジェスチャの指示内容のタイプに基づいている(表2参照)。

表2　語句と指示内容の対応

語句	格	対応する指示内容のタイプ
(この)場所	述語一般/LOCT, *COPULA*/IDEN, *COPULA*/OBJE	MOBJ
	BE-LOCATED/LOCT	MPART
(この)あたり(辺、部分)		MPART
(この)+普通/固有名詞		MOBJ
こちら	述語一般/LOCT, *COPULA*/IDEN, COPULA*/OBJE	MOBJ
	BE-LOCATED/LOCT	MPART
これ		MOBJ
ここ	述語一般/LOCT, *COPULA*/IDEN, *COPULA*/OBJE	MOBJ
	BE-LOCATED/LOCT	MPART
普通/固有名詞		MOBJ

3.7.1.3の例では、ジェスチャ対応語句が「こちら」で、さらにそれは*BE-LOCATED*/LOCTなので、ジェスチャの指示内容候補

[MOBJ [[ID n京都ホテル]]]
[MOBJ [[ID n京都府庁]]]
[MOBJ [[ID n御池通り]]]
[MOBJ [[ID n中京区]]]
[MOBJ [[ID n地図]]]
[MPART [[ID n中京区][MAREA [[X1 x1][Y1 y1][X2 x2][Y2 y2]]]]]
[MPART [[ID n地図][MAREA [[X1 x1'][Y1 y1'][X2 x2'][Y2 y2']]]]]

から、最終的には、

[MPART [[ID n中京区][MAREA [[X1 x1][Y1 y1][X2 x2][Y2 y2]]]]]

が「こちら」に対応するジェスチャ情報として選択される。

なお、ポインティング、スクランブリング、マーキングに関しても同様に対応を同定するので説明を省略する。

3.7.2　ライン・ドラッギング・ジェスチャと発話の対応関係の同定

ライン・ドラッギング・ジェスチャの指示内容の表現は、3.6.2.2.2のように、意味的に分割された

状態である。この分割された各要素(START、MOVE、TURN、END)をここでは断片ジェスチャ情報と呼ぶことにする。ライン・ドラッギング・ジェスチャと発話の対応関係の同定は、この断片ジェスチャ情報と発話の述語の単位との対応を同定し、さらに、前者の内部的な情報(FROM、TO、PASSなどの情報)と後者の内部的な情報(DEPT、DEST、ROUTなどの情報)の対応を同定することによって行われる。

3.7.2.1　対応同定の例外処理

「ここを曲がってここまで行ってください」という発話と、一度曲がるライン・ドラッギング・ジェスチャ(図10のような)の組み合わせを考えると、[TURN[[AT 四条川端][DIRECTION 270]]]と「ここを曲がる」という対応と、[MOVE[[FROM 四条川端][TO 京阪三条]]]と「ここまで行く」という対応にみられるように、通常、ひとつの断片ジェスチャ情報は、発話のひとつの命題の単位に対応すると考えれらる。しかし、「ここまで行ってください」という発話が、道路上のライン・ドラッギング・ジェスチャとともになされた場合は、「ここまで行く」というひとつの命題の単位に、すべての断片ジェスチャ情報が対応すると考えられる。このように、断片ジェスチャ情報と発話の命題の単位の一対一の対応が崩れる場合もありうるので、まずはじめにそれらの対応を例外的に処理することにする。

(1)「ここまで行ってください」という発話のように、

行く、進む

(DEPT)

DEST

という命題内容をもつ述語の単位が単独で存在する場合、その単位と対応するジェスチャ情報は、すべての断片ジェスチャ情報であると同定し、DEPTとDESTはそれぞれ、断片ジェスチャ情報のSTARTとENDの値に対応すると判断する。

2)「ここで曲がって、ここまで行ってください」という発話のように、

行く、進む

DEST

という命題内容をもつ述語の単位が述語の並列構造の最後尾に存在する場合、その単位と対応するジェスチャ情報は、行為ラベルがENDである断片ジェスチャ情報であると同定し、DESTは、断片ジェスチャ情報のENDの値に対応すると判断する。

3)「このように行ってくさだい」という発話のように、

行く、進む

MANN　このように(こう)

という命題内容をもつ述語の単位が単独で存在する場合、その単位と対応するジェスチャ情報は、すべての断片ジェスチャ情報であると同定し、MANNは、すべての断片ジェスチャ情報に対応すると判断する。

3.7.2.2　発話情報の述語　/格要素とジェスチャのラベルとの対応

前項以外の場合については、次ページの表3の対応パターンに基づいて対応同定を行う。

表3　発話とジェスチャの対応パターン

発話中の述語と格	対応するジェスチャのラベル
出る OBJE(を格)	START STARTの値
出る DEPT(から格)	START STARTの値
出る DEST(に格、まで格)	END ENDの値
行く、歩く、進む DEST(まで格、に格)	MOVE TOの値
行く、歩く、進む ROUT(を格)	MOVE ALONGの値
通る ROUT(を格)	MOVE PASSの値、ALONGの値
通り過ぎる ROUT(を格)	MOVE PASSの値
曲がる LOCT(で格、を格)	TURN ATの値
曲がる OBJE(に格)	TURN DIRECTIONの値
曲がる MANN	TURN DIRECTIONの値

3.7.2.3　対応同定例

図10に示すジェスチャが行われたと仮定し、以下では、そのジェスチャと同時に発話される文の例をふたつあげ、それぞれの場合について対応同定の処理のようすを示す。

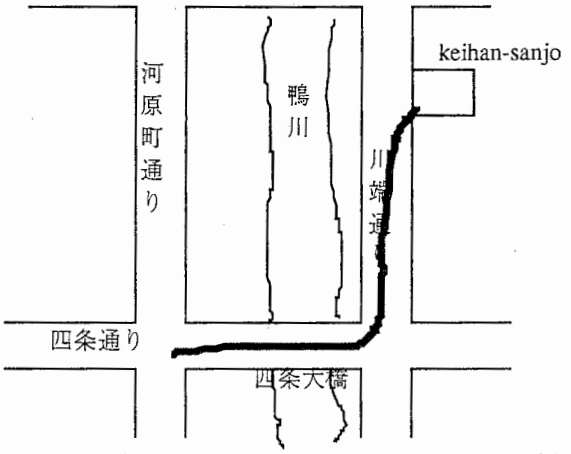


図10　ライン・ドラッギング・ジェスチャの例

[[START [MOBJ n四条河原町]]
[MOVE [[FROM [MOBJ n四条河原町]]
[TO [MOBJ n四条川端]]
[ALONG [MOBJ n四条通り]]
[PASS [MOBJ n四条大橋]]]
[TURN [[AT [MOBJ n四条川端]]
[DIRECTION 270]]
[MOVE [[FROM [MOBJ n四条川端]]
[TO [MOBJ n京阪三条]]
[ALONG [MOBJ n川端通り]]
[PASS NULL]]
[END [MOBJ n京阪三条]]]

図11　ジェスチャの意味表現

(例1)
発話：「ここで左に曲がって、川端通りを真っ直ぐ進んでください」
発話の意味表現：
[SEM [[RELN 並列表現]
[ARG1 [[RELN *ACTION-REQUEST*]
[AGEN *SPEAKER*]
[RECP *HEARER*]
[OBJE [[RELN *曲がる*]
[SUBJ *HEARER*]
[LOCT [[RELN *ここ*]]]
[OBJE [[RELN *左*]]]]]]]
[ARG2 [[RELN *ACTION-REQUEST*]
[AGEN *SPEAKER*]
[RECP *HEARER*]
[OBJE [[RELN *進む*]
[SUBJ *HEARER*]
[ROUT [[RELN *川端通り*]]]
[MANN [[RELN *真っ直ぐ*]]]]]]]]

発話の意味表現中の、
曲がる
LOCT　ここ
OBJE　左
に対応する断片ジェスチャ情報として、ジェスチャ情報中の、
TURN
AT [MOBJ n四条川端]
DIRECTION 270
が同定され、「ここ」が「四条川端」であることがわかる。
次の、
進む
ROUT　川端通り
MANN　真っ直ぐ
には、

MOVE
FROM　四条川端
TO　　京阪三条
ALONG　川端通り
PASS　　NULL
が対応する。

(例2)
発話：「ホテルを出て、このように行って下さい」
発話の意味表現：
[SEM [[RELN 並列表現]
[ARG1 [[RELN *ACTION-REQUEST*]
[AGEN *SPEAKER*]
[RECP *HEARER*]
[OBJE [[RELN *出る*]
[SUBJ *HEARER*]
[OBJE [[RELN *ホテル*]]]]]]]
[ARG2 [[RELN *ACTION-REQUEST*]
[AGEN *SPEAKER*]
[RECP *HEARER*]
[OBJE [[RELN *行く*]
[SUBJ *HEARER*]
[MANN [[RELN *このように*]]]]]]]]

発話の意味表現中の、
出る
OBJE
という構造は、ジェスチャ情報中の、
[START [MOBJ n四条河原町]]
との対応が同定され、
行く
MANN　このように
には、ジェスチャ構造の2番目の断片ジェスチャ情報、
[MOVE [[FROM [MOBJ n四条河原町]]
[TO [MOBJ n四条川端]]
[ALONG [MOBJ n四条通り]]
[PASS [MOBJ n四条大橋]]]
以下のすべての断片ジェスチャ情報が対応する。

4　マルチモーダル問い合わせシステムの試作

以上で述べた、音声とジェスチャを統合的に意味解析するインターフェースを用いて、道案内に関する簡単な問い合わせシステムを試作した。このシステムは、地図上のオブジェクトや道順に関する、ユーザの13の問い合わせ発話(付録4)と5種類のジェスチャの統合的な意味を同定した後、それに対する適切な応答を生成し、合成音声と地図上のサークリングやラインドラッギングの描画によってユーザに情報を提供する。使用方法は付録1に添付する。本章では、統合的意味が同定された後の応答生

成処理について簡単に述べる。

4.1 応答生成処理

応答生成処理部には、発話とジェスチャの統合意味解析結果が入力され、発話の意味表現のパターンに基づいて、データベースの検索などの必要な処理が行われるとともに、応答文が生成される(表4参照)。

表4 意味表現のパターンに対する処理と応答文

発話の意味表現	処理内容	応答文
INFORM DESIRE 見る OBJ	Video(OBJ) #OBJに関する動画を再生する	このようになっています
INFORM DESIRE 行く DEST	Path(Current, Destination) #CurrentとDestinationの間の最短経路を探す	このように行ってください
WH-QUESTION INFORMREF どちら RESTR 安い COMP ここ ここ	min(Price(A), Price(B)) #AとBで値段が安い方を探す	こちらの…が安いです
WH-QUESTION INFORMREF 何 RESTR これ	Name(OBJ) #OBJの名前を調べる	これは…です
YN-QUESTION BE-LOCATED OBJ 安いホテル LOCT このあたり(MPART)	min(Price(A), Price(B), …)	このホテルがいちばん安いです
YN-QUESTION BE-LOCATED OBJ 部屋 LOCT 都ホテル(MOBJ)	Reserved(OBJ) #OBJに空部屋があるかどうか調べる	はいあります ／いいえありません
YN-QUESTION BE-LOCATED OBJ 部屋 LOCT このホテル(MOBJ)	Reserved(OBJ)	はいあります ／いいえありません

YN-QUESTION COPULA IDEN これ OBJE ホテル	Property(OBJ) #OBJの属性を調べる	はいそうです ／いいえこれは…です
CONFIRMATION-Q 行く DEST MANN	Path(Departure, Destination) ∋ Gestre(このように) #なされたジェスチャが最短経路かどうか調べる	はいそうです ／いいえこのように行ってください
CONFIRMATION-Q 出る DEPT 行く MANN このように	Path(Departure, Destination) ∋ Gestre(このように)	はいそうです ／いいえこのように行ってください
CONFIRMATION-Q 曲がる LOCT ここ OBJE 左 DEST	Path(Departure, Destination) ∋ Gestre	はいそうです ／いいえこのように行ってください

システムの動作の例をあげると、「ここととここではどちらが安いですか」という発話とともに、Aホテルと Bホテルにサークリング・ジェスチャを行った場合、統合的意味を同定した後、データベースのホテルの値段を検索して安い方のホテルを調べ、「こちらのAホテル(このときAホテルが丸で囲まれる)が安いです」という応答を返す。

5　おわりに

本稿では、入力された音声とポインティング・ジェスチャの統合的な意味を同定する枠組みについて述べた。さらに、その枠組を応用して試作した、マルチモーダル問い合わせシステムについても簡単に言及した。この解析の枠組みでは、ジェスチャ側の情報は、発話の意味解析後にはじめて考慮されて意味統合される形をとっている。しかしながら、ジェスチャの情報は、意味解析以前の言語解析、すなわち、構文解析、形態素解析、音声認識のレベルで解析精度などの向上に貢献できる可能性も考えられるので、今後はその点について検討してゆく。

参考文献

[1]M. Blattner and R. Dannenberg (Ed.), "Multimedeia Interface Design," ACM Press, 1992
[2]水梨, "マルチモーダル・ヒューマン・コンピュータ・インタラクション関連文献調査," ATRテクニカルレポート, TR-IT-0060, 1994.6
[3]ローケン・キム, 谷戸, 森元, "マルチモーダル音声翻訳通信のためのシミュレータ" 情報処理学会オーディオビジュアル複合情報処理研究会4-16(94.3.18)
[4]水梨, ローケン・キム, パク, 友清, 森元, "マルチモーダル翻訳対話における発話とジェスチャの関連性," 情報処理学会音声言語情報処理研究会, 95-SLP-6, 1995.5
[5]Park, Loken-Kim, "Text Database of the Telephone and Multimedia Multimodal Interpretation Experiment," ATRテクニカルレポート, TR-IT-0086, 1994.12
[6]Park, Loken-Kim, Mizunashi, Fais, "Transcription of the Collected Dialogue in a Telephone and Multimedia and Multimedia/Multimodal WOZ Experiment," ATRテクニカルレポート, TR-IT-0090, 1995.2
[7]T. Shimizu, et al, "Spontaneous dialogue speech recognition using cross-word context constrained word graphs," Proc. of ICASSP'96,1996.
[8]田代他, "音声言語処理のための構文解析ツールキット," 情報処理学会研究会報告NL-106-12, 1995
[9]永田, 田代他, "日本語解析文法　解説書," ATRテクニカルレポート, TR-I-0335, 1993.3

付録1　問い合わせシステムの使用法

システム動作環境

ハードウェア

Sun SparcStation 20 タッチパネル仕様 (例えばblux)
Sun SparcStation 10 (例えばgamba)
DATLINK
DATデッキ

ソフトウェア

ソフトウェア一式は blux:/home/blux/MMDS/GES/Ges9610 (以降 $MMDS と表記)の下にある。

1) 音声認識 SPRECシステム一式 $MMDS/SPREC
2) 音声合成 Chatrシステム一式 /DB/PI/bin
3) 構文解析 MMparse.lisp システム一式 $MMDS/PARSER
4) システム本体 $MMDS/BIN
   Gesture_Prog : システム全体のシーケンスコントロールとユーザ・インターフェースのプログラム
   unifier : 発話とジェスチャの意味構造を統合するプログラム
5) その他関連ソフトウェア(オリジナル) $MMDS/BIN
   sptee : UIプログラムの PedalDown指示によって音声入力のon/offを行なうプログラム
   sp2pa : 音声認識プログラムの出力と構文解析プログラムの入力のあいだでフォーマットを変換するプログラム
6) その他関連ソフトウェア(既製) 提供ディレクトリ
   narecord : DATLINK からの音声入力を計算機に取り込むプログラム(DATLINK提供)
   raw2audio : 音声生データをSun au フォーマットに変換(Sun提供)
   play : au フォーマット音声データをスピーカから再生するプログラム(Sun提供)
7) データファイル $MMDS/BIN
   MapDB.dat : マップデータベースファイル
   Shizishi.dat : 指示詞データファイル
   Noun.dat : 名詞データファイル
   Adjective.dat : 形容詞データファイル

システム構築方法

以下にモジュールの構築に必要な情報を述べる。

1) ライブラリの作成
   $MMDS/Geslibで make を実行することによってライブラリ libges.a が生成される。
2) 統合モジュールの作成
   $MMDS/Unifyで make を実行することによって統合モジュール unifier が生成される。
3) メインモジュール(ユーザーインターフェイスを含む)の構築
   $MMDS/UI で make を実行することによってメインモジュール Gesture_Prog が生成される。
4) 関連ソフトウエアの作成
   $MMDS/TOOLで以下のコマンドを入力する。

```
% cc -o ../BIN/sp2pa sp2pa.c
% cc -o ../BIN/sptee sptee.c
```

実行方法

本システムは以下の手順にしたがって動作させることができる。

1) 起動ターミナルの準備
本システムは大きくわけて3つのサブシステムで構成されている。現時点ではその3システムの起動は連携されておらず、使用者が別々のターミナルからそれぞれのシステムを起動する必要がある。したがって、使用者は最初に画面上に3つのターミナルをオープンしなければならない。

2) 各サブシステムの起動
各プログラムの起動は以下のコマンドをそれぞれのターミナルから入力する。

音声認識プログラム起動

```
% chatr-0.9 --server
```

音声合成プログラム起動

```
% setenv ATRSPREC /usr/local/SPREC/r04a04 % narecord -o right -s 11025
  | sptee | sprec -config=arg |& sp2pa
```

本体の起動

```
% Gesture_Prog
```

3) デモンストレーションの実施
画面右下の PEDAL ボタンにタッチする。
例文の発話を行ないながらジェスチャを入力する。
もう一度 PEDAL ボタンにタッチする。

付録2　プログラム解説

意味統合システムのプログラム中、ジェスチャ解析部、意味統合部、地図データベース部に実装された各クラスの概要を示す。

ジェスチャ解析部クラス

class Point 座標情報クラス
スーパークラス なし
データ
double px; X座標
double py; Y座標
メソッド
double x(void); X座標
double y(void); Y座標
double distance(Point & ip); 2点間の距離
Point center(Point p); 2点の中点

class Line 線分情報クラス
スーパークラス なし
データ
Point p0; 始点座標
Point p1; 終点座標
メソッド
Point getp0(void); 始点
Point getp1(void); 終点
Bool and(Line, Point &); 2線分の交点
distance(Point &); 2線分の距離

class Rect 矩形情報クラス
スーパークラス なし
データ
Point p0; 左上点
Point p1; 右下点
メソッド
Point getp0(void); 左上点
Point getp1(void); 右下点
Bool exist(Point); ある点が矩形内部に存在するか
Bool and(Rect, Rect &); 2矩形が交わるか
Bool and(Line, Line &); 矩形と線分が交わるか
Point center(void); 矩形の中心点
double square(void); 矩形の面積

class PointWithTime 座標時間情報クラス
スーパークラス Point
データ
Point p; 座標情報

Usl t; 時間情報
メソッド
なし

class PointWithTimes 座標時間情報群クラス
スーパークラス なし
データ
PointWithTime *points; 座標時間情報群
Usl num; 包含個数
Usl size; バッファサイズ
メソッド
Usl addPoint(PointWithTime &); 座標時間情報を登録

class Gesture ジェスチャ情報クラス
スーパークラス なし
データ
int type; ジェスチャタイプ
int stime; ジェスチャ開始時間
int etime; ジェスチャ終了時間
メソッド
int setType(int tp); ジェスチャタイプ設定
int getType(); ジェスチャタイプ取得
void draw(int mode=1); ジェスチャ再生

class TimeAndCoord 座標時間情報クラス
スーパークラス なし
データ
int x; X座標
int y; Y座標
int tm; 時間情報
メソッド
void setPoint(int ix=0, int iy=0, int tm=0); データ設定
void setPoint(TimeAndCoord tc); データ設定
double slope(TimeAndCoord tc); 2点の変化率(傾き)
double distance(int ix, int iy); 2点間の距離
double distanceL(float a, float b, float c); 2点間の距離
void line1(TimeAndCoord tc, float *a, float *b, float *c); 2点を結ぶ線分
void line1(TimeAndCoord tc1, TimeAndCoord tc2, float *a, float *b, float *c); 2点を結ぶ線分
void findObject(char *line); 地図オブジェクトサーチ
void findRoadObject(char *name); 地図(道路)オブジェクトサーチ
void findPassObject(char *name); 地図(通過)オブジェクトサーチ
void draw(int c=1); 点の描画
void drawLine(TimeAndCoord tc, int mode=1); 2点間線分の描画 int tellx(); X座標取得
int telly(); Y座標取得
void print(FILE *); 構造化情報出力

class Henkaten 変化点情報クラス
スーパークラス なし
データ
TimeAndCoord *coord; 座標時間情報
int attr; 属性(スタートなど)
char object[64]; 検出地図オブジェクト名称
char object_id; 地図オブジェクトID
メソッド
void findObject(); 地図オブジェクト検出
void print(FILE *); 構造化情報出力

class Kukan 区間情報クラス
スーパークラス なし
データ
TimeAndCoord *coord; 座標時間情報群
int num; 座標時間情報個数
int size; バッファサイズ
char object[64]; 検出地図オブジェクト名称
char pass[20][64]; 検出地図(通過)オブジェクト名称
int pass_num; 通過オブジェクト個数
メソッド
void addPoint(TimeAndCoord tc); 座標時間情報追加
void findObject(); 地図オブジェクト検出
void draw(int mode); 区間再生描画
void print(FILE *); 構造化情報出力

class CirclingGesture サークリングジェスチャ情報
スーパークラス Gesture
データ
TimeAndCoord* coords; 座標時間情報群
int num; 座標時間情報個数
int size; バッファサイズ
int max_x; 最大X座標
int min_x; 最小X座標
int max_y; 最大Y座標
int min_y; 最小Y座標
メソッド
void addPoint(TimeAndCoord tc); 座標時間情報追加
void draw(int mode=1); ジェスチャ再生
void print(FILE *); 構造化情報出力
void setMaxMin(int ax, int ix, int ay, int iy); 最大最小値設定

class PointingGesture ポインティングジェスチャ情報
スーパークラス Gesture
データ
Point g; 座標情報
メソッド

void setg(Point & ig); 座標設定
void print(FILE *); 構造化情報出力
void draw(int mode=1); ポインティングジェスチャ再生

class DraggingGesture ラインドラッギングジェスチャ情報
スーパークラス Gesture
データ
int numHenka; 変化点個数
Henkaten *henka[MAX_HENKA]; 変化点情報
Kukan *kukan[MAX_KUKAN]; 区間情報
メソッド
void addPoint(TimeAndCoord tc, int attr); 座標時間情報追加
void addKukanPoint(TimeAndCoord tc); 区間に座標時間情報追加
void findObject(); 地図オブジェクト検出
void print(FILE *); 構造化情報出力
void draw(int mode=1); ラインドラッギングジェスチャ再生

class GestureAnalysis ジェスチャ解析クラス
スーパークラス なし
データ
TimeAndCoord *coord; 認識対象座標時間群
int num; 有効座標数
int size; 確保領域サイズ
メソッド
void addPoint(int ix=0, int iy=0, int it=0); 座標時間情報追加
Gesture* analysis(PointWithTimes &); ジェスチャ解析
Gesture* analysis3(); ジェスチャ解析2
Gesture* analysisCircle(); サークリングジェスチャ解析
Gesture* analysisPoint(); ポインティングジェスチャ解析
DraggingGesture *isLine(DraggingGesture *dg, int s, int e); ラインドラッギング
ジェスチャ解析

意味統合部クラス

class FsValue 意味構造基底クラス
スーパークラス なし
データ
int type; 構造タイプ
継承メソッド
void delete_object(FsValue*); オブジェクトの削除
FsValue* create_object(FsValue*); オブジェクトの生成 メソッド
int getType(void); 構造タイプの取得

class FsValues 意味構造群クラス
スーパークラス なし
データ
FsValue** values; 意味構造群

int num_value; 意味構造個数
メソッド
void addValue(FsValue*); 意味構造追加
void addValue(FsValues*); 意味構造群追加
FsValue* searchValue(int i = 0); n 番めのノード取得
void print(void); 構造データ出力
int getNumOfItem(void); 意味構造個数の取得
FsValues* selectDraggingGesture(void); ラインドラッギングジェスチャ意味構造検索
FsValues* selectOtherGesture(void); その他ジェスチャの意味構造検索

class FsBasic 意味構造(基本形)クラス
スーパークラス FsValue
データ
char attr[20]; タグ(attribute)
FsValue *value; 値(value)
メソッド
char* getAttr(void); タグの取得
FsValue* getValue(); 値の取得
FsValue* searchValue(const char* ="", int=1); 指定TAG ノードの検索
void print(int=0); 構造データの出力
int searchContent(const char*); 指定content ノードの検索
int cmpString(const char*); content 比較
int kakaruNode(FsValue* &); かかりうけノードの検索

class FsPrim 意味構造(プリミティブ)クラス
スーパークラス FsValue
データ
char *value; 値
メソッド
int cmpValue(const char*); 値の比較
void copyString(char* &); 値の取得
void print_id(int=0); 構造データの出力
FsValue* searchValue(const char* ="", int=1); 指定TAG ノードの検索
void print(int=0); 構造データの出力
int searchContent(const char*); 指定content ノードの検索
int cmpString(const char*); content 比較
int kakaruNode(FsValue* &); かかりうけノードの検索

class FsSet 意味構造(集合)クラス
スーパークラス FsValue
データ
FsValue **values; 意味構造集合
int num; 集合個数
メソッド
void setValue(FsValue*); 集合に意味構造を加える
FsValue* searchValue(const char* ="", int=1); 指定TAG ノードの検索
FsValue* searchValue(int i); n 番めの意味構造取得

void deleteValue(int); n 番めの意味構造削除
void deleteValue(const char*); TAG指定で意味構造削除
void print(int=0); 構造データの出力
int searchContent(const char*); 指定content ノードの検索
int cmpString(const char*); content 比較
int kakaruNode(FsValue* &); かかりうけノードの検索

class BuildFetureStructure 意味構造作成クラス
スーパークラス なし
データ
char *content; 処理文字列
long size; 文字列サイズ
long ptr; 処理ポインタ
メソッド
FsValues* build(const char* = NULL); 意味構造作成

class Kakuyouso 格要素情報クラス
スーパークラス なし
データ
char *tag; 格要素タグ
FsValue *value; 発話意味構造
int parallel; 並列情報フラグ
FsValue *gesture; 結合ジェスチャ意味構造 メソッド
void print(void); 構造化データ出力
int searchShizishi(void); 格要素のなかに指示詞が存在するか
FsValue* searchValue(const char* str); 指定TAGの意味構造をとり出す
int cmpString(const char *str); タグと文字列の比較
void addGesture(FsValue*); 格要素にかかるジェスチャの登録
int existGesture(void); かかるジェスチャの存在チェック
void selectGesMpart(void); ジェスチャのMPARTだけを抽出
void selectGesMobj(void); ジェスチャのMOBJだけを抽出
void selectGesCat(int); ジェスチャのカテゴリ該当だけ抽出
FsValue* searchGesture(void); ジェスチャ情報の取り出し
int checkParallel(void); 並列構造か検査
FsValue* Kakuyouso::kakaruNode(void); かかり受けノードの検出

class ZyutsugoUnit 述語単位クラス
スーパークラスなし
データ
char *zyutsugo; 述語
Kakuyouso **kakuyouso; かかる格要素
int num_youso; 格要素数
FsValue *gesture; 述語単位にかかるジェスチャ情報
メソッド
void addGesture(const int , FsValues*, const int); ジェスチャの登録
void print(void); 述語単位情報出力
inline int getNumOfItem(void); 格要素数の取得

inline int cmpString(const char *str); 述語と文字列の比較
inline void copyString(char *str); 述語の取得
int searchShizishi(void); 述語単位に指示詞を含むか
inline Kakuyouso* searchValue(int i=0); 格要素の取り出し
Kakuyouso* searchValue(const char*); 格要素の取り出し
Kakuyouso* searchGestureKaku(void); ジェスチャ付き格要素の取り出し
inline FsValue* searchGesture(void); ジェスチャ情報の取り出し
void selectGesMpart(void); ジェスチャのMPARTだけを抽出
void selectGesMobj(void); ジェスチャのMOBJだけを抽出
void optiForCircle(void); サークリング用最適化

class ZyutsugoUnits 述語単位群クラス
スーパークラス なし
データ
ZyutsugoUnit **units; 述語単位群
long num_unit; 述語単位数
メソッド
void addUnit(ZyutsugoUnit*); 述語単位の追加
inline void print(void); 述語単位群情報の出力
ZyutsugoUnit* searchValue(long i); 述語単位のとりだし
inline long getNumOfItem(void); 述語単位数の取り出し
inline void optiForCircle(void); サークリングジェスチャ用最適化

class HatsuwaKouzou 発話構造情報クラス
スーパークラス なし
データ
FsValues *values; 発話情報
メソッド
ZyutsugoUnits* selectZyutsugo(void); 述語検索
inline void print(void); 発話構造情報の出力

class BuildHatsuwaKouzou 発話構造構築クラス
スーパークラス BuildFetureStructure
データ
なし
メソッド
HatsuwaKouzou* build(const char*); 発話構造情報構築

class Unifier 発話/ジェスチャ統合クラス
スーパークラス なし
データ
なし
メソッド
ZyutsugoUnits *unify(FsValue* , FsValues* &); 情報統合

class Solver 問題解決クラス
スーパークラス なし

データ
char reply[1024]; 応答文字列
メソッド
int pattern(FsValue*, ZyutsugoUnits*); 応答パターン選択
void outou(int, ZyutsugoUnits*, FILE *); 応答文生成
void play(void); 応答文字列再生

地図データベース部クラス

class MapObj マップオブジェクト情報クラス
スーパークラス なし
データ
Usl id; オブジェクトID
Uss category; オブジェクトカテゴリ
char name[40]; オブジェクト名称
メソッド
Uss getCategory(void); カテゴリ取得
Bool isName(const char* iname); 名称照合
Bool isId(const int iid); ID照合
Bool isCategory(Uss cat); カテゴリ照合
void print(FILE *); 構造化情報(IDのみ)出力
void print_name(FILE *); 構造化情報(名前つき)出力
char *getName(void); 名称取得

class PointMapObj 点型地図オブジェクトクラス
スーパークラス MapObj
データ
int price; 値段
int reserve; 予約フラグ
Rect area; オブジェクト領域
Point g; オブジェクト代表位置
メソッド
Bool isInclude(Point); 点がオブジェクトに包含されるか
Bool isRectRelation(Rect, Rect &); 矩形との関係
Bool isSimilar(Rect); 同じ領域か
double distance(Point); 点とオブジェクトの距離
Bool isReserved(void); (ホテルが)予約されているか
int getPrice(void); (ホテルの)値段を取得
Rect* getRect(void); オブジェクト領域の取得

class LineMapObj 線型地図オブジェクトクラス
スーパークラス MapObj
データ
Line line; 代表線
Uss w; 線幅
メソッド
Bool isInclude(Point); 点がオブジェクトに包含されるか

Bool isRectRelation(Rect, Rect &); 矩形との関係
Bool isSimilar(Rect); 同じ領域か
double distance(Point); 点とオブジェクトの距離
Bool isReserved(void); (ホテルが)予約されているか
int getPrice(void); (ホテルの)値段を取得
Rect* getRect(void); オブジェクト領域の取得

class MapDB 地図データベース処理クラス
スーパークラス なし
データ
MapObj* object[MaxMapObject]; 登録オブジェクトバッファ
Usl num_object; 登録オブジェクト個数
MapObj* selection[MaxMapObject]; 選択オブジェクトバッファ
Usl num_selection; 選択オブジェクト個数
MapObj* selection2[MaxMapObject]; 選択オブジェクト2バッファ
Usl num_selection2; 選択オブジェクト2個数
メソッド
void selectAll(void); データベース初期化
void selectCategory(Uss cat); カテゴリで絞り込み
void selectRect(Rect); 領域で絞り込み
void sortDistance(Point); 点からの距離順にソート
void selectMinDistance(Point); 点から一番近いものに絞り込み
void selectInclude(Point); 点を包含するものを絞り込み
void selectSimilar(Rect); 領域で絞り込み
void withoutSimilar(Rect); 領域を除いて絞り込み
void selectCheep(void); 最も安いものに絞り込み
MapObj* searchByName(const char*); 名称で絞り込み
MapObj* searchById(const int); IDで絞り込み
Usl countSelection(void); 選択オブジェクト個数の取得
Usl countSelection2(void); 選択オブジェクト2個数の取得
MapObj* getSelection(void); 選択オブジェクトの取得

付録3　意味統合システムの入力想定文

ここで左に曲がって川端通りを真っ直ぐ進んでください
こちらのほうから出て川端通りを通ってください
ホテルを出てこのように行ってください
バスで三条京阪まで行ってください
このホテルまで行ってください
タクシー乗り場はここにあります
四条大橋を通り過ぎてここまで行ってください
ここで乗り換えて二つ目の駅で降りてください
ここで曲がって二つ目の角でまた左に曲がってください
今あなたはこの場所にいます
ここになります
ここにあります
こちらが都ホテルになります
あなたは今ここにいます
あなたは今ここですね
ここが国際会議センターです
この丸のところに着きます
タクシー乗り場はここです
ここからタクシーに乗ってください
このあたりです
京津線に乗って二つ目の蹴上駅で降りてください
ここからここまでこのように行ってください
このホテルは高いですがこちらは安いです
ここととここは安いです

付録4　問い合わせシステムの入力想定文

このホテルの中の様子を見たいのですが
ここととここではどちらが安いですか
このあたりに安いホテルはありますか
これは何ですか
この黄色い建物は何ですか
これはホテルですか
都ホテルには部屋はありますか
このホテルには部屋はありますか
このホテルに行きたいのですが
このあたりを拡大してください
このホテルへはこのように行くんですね
三条京阪へはホテルを出てこのように行くんですね
このホテルに行くためにはここで左に曲がるんですね