環境シミュレータ RTC
取扱説明書

豊橋技術科学大学 行動知能システム学研究室

平成 24 年 3 月 25 日

目 次

1　初めに

このドキュメントでは環境シミュレータ RTC について解説し，その使い方を説明します.

1.1　開発・動作環境

環境シミュレータ RTC は以下の環境で開発し，動作確認を行っています.

- Windows XP Pro SP3
- Open-rtm-aist 1.0.0(C++版)
- Visual studio 2008

また，OpenCV 2.1 および CGAL 3.7 を開発に使用しています．OpenCV については下記サイトを参照して下さい.
http://sourceforge.net/projects/opencvlibrary/

CGAL(Computational Geometry Algorithms Library) については下記サイトを参照して下さい.
http://www.cgal.org/

2　環境シミュレータ RTC について

この章では，環境シミュレータの概要と，RTC の持っている各ポートおよびコンフィグレーションについて説明します．図 1 は環境シミュレータ RTC の外観です.

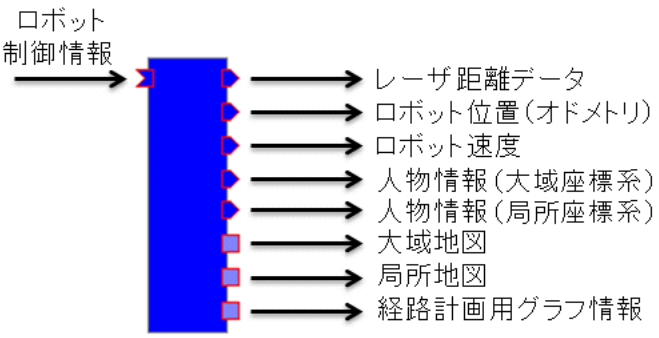

図 1: 環境シミュレータ RTC の外観

2.1 概要

環境シミュレータは屋内環境，そこで行動するロボット，そして環境内の人の動きを再現します．歩行者の動きは「大域的な動き」および「個人レベルの局所的な動き」に分類され，シミュレータではポテンシャルモデルと移動ネットワークを組み合わせる事によってこれを再現しています．また，障害物（壁）の情報と人の行動に関する情報が記録された環境データはテキストファイルとして用意します．障害物の情報は 2 つの座標を結ぶ線分として表現されており，線分の集合として表現することができればある程度複雑な環境でも再現することができます．環境データのファイル名をシミュレータ RTC のコンフィグレーションで指定することにより，簡単に環境を切り替えてシミュレーションを行う事ができます．

環境シミュレータの画面を図 2 に示します．この図の環境は食堂をモデル化し，再現したものです．この図中ではロボットは橙色の丸で表され，人は緑色の丸で表されています．

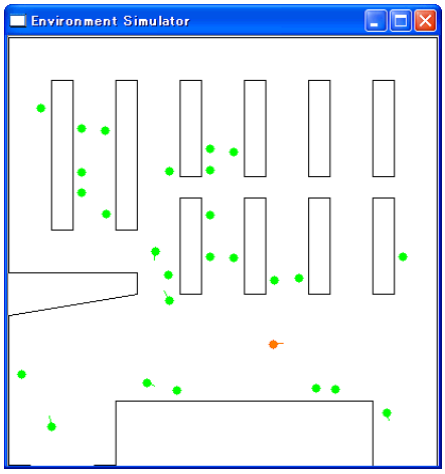

図 2: シミュレーションの様子

また，シミュレータは人の移動に加え,「席で休む」,「行列に並ぶ」といった公共空間で見られる特徴的な人の行動もまたモデル化し，再現しています．

2.2 データポート

環境シミュレータ RTC が持つデータポートの一覧を表 1 に示します.

表 1: 入出力データポート

Port Type	Data Type	Port Name	備考
In Port	IIS::TimedVelocity2D	robotControlInfo	ロボット制御命令
Out Port	SensorRTC::LaserRangeSensor:: idl::TimedMeasuredData	rangedata	レーザ距離データの出力
Out Port	IIS::TimedPose2D	robot_state	ロボットのオドメトリ（大域座標）
Out Port	IIS::TimedVelocity2D	robot_velocity	ロボットの速度情報
Out Port	TUT::TimedPersonsGlobalPosition	persons	絶対座標系人物情報
Out Port	TimedPeopleTrackingData	peopledata	ロボット座標系人物情報

2.3 サービスポート

サービスポートの一覧を表 2 に示します.

表 2: サービスポート

Port Type	Data Type	Port Name	備考
Service Provider	MRFC:: AbsoluteMapService	AbsoluteMapService	大域地図（絶対座標系地図） の出力
Service Provider	MRFC:: RelativeMapService	RelativeMapService	局所地図（ロボット座標系地図） の出力
Service Provider	TUT::AreaInfoService	AreaInfoService	経路計画用のグラフ情報の出力

2.4 コンフィグレーション

この節では，環境シミュレータ RTC が持つコンフィグレーションについて説明します．

2.4.1 FieldDataFile

表 3: 環境データを指定するコンフィグレーション

型	変数名	単位
std::string	FieldDataFile	-

FieldDataFile でシミュレーションに用いる環境のデータを指定します．環境データには障害物（壁）の情報と人の行動に関する情報が記録されている．環境データはテキストファイルとして用意します．障害物の情報は 2 つの座標を結ぶ線分として表現されており，線分の集合として表現することができればある程度複雑な環境でも再現することができます．環境データのファイル名をこのコンフィグレーションで指定することにより，簡単に環境を切り替えてシミュレーションを行う事ができます．

この環境データの仕様については後述します．

2.4.2 walkingrate_mean，walkingrate_std_deviation

表 4: 人の歩行速度に関するコンフィグレーション

型	変数名	単位
double	walkingrate_mean	m/s
double	walkingrate_std_deviation	m/s

シミュレータ上で再現された人の歩行速度は正規分布に基いたばらつきを持っており，その値はこの 2 つのコンフィグレーションの値で決定します．walkingrate_mean は歩行速度の平均，walkingrate_std_deviation は歩行速度の標準偏差です．

2.4.3 localmap_width，localmap_height，localmap_scale，globalmap_scale

表 5: 出力する地図関係のコンフィグレーション

型	変数名	単位
int	localmap_width	cell
int	localmap_height	cell
double	localmap_scale	m/cell
double	globalmap_scale	m/cell

環境シミュレータ RTC はサービスポートを通じて 2 種類の地図を出力します．これらのコンフィグレーションはその地図についての設定を行います．lcalmap_width および localmap_height は出力する局所地図（ロボット座標系地図）の縦横の大きさを決めます．また，localmap_scale は局所地図のスケール（1cell のサイズ）を決めます．同様に，globalmap_scale は出力する大域地図（絶対座標系地図）のスケールを決めます．

2.4.4 following_target

表 6: 人物情報のコンフィグレーション

型	変数名	単位
int	following_target_id	-

環境シミュレータ RTC が出力する人物情報は，ロボットが特定の人物追従を行うことを想定したものとなっており，環境内のどの人物を追従対象とするのかを指定することができます．そこで，環境内に現れる何番目の人物を追従対象として指定するのかを following_target_id で決めることができます．ここで，この値は最初に出現する人物を 0 として指定します．また，追従対象を設定しない場合は負数を指定します．

2.4.5 simulate_people, simulate_skip

表 7: 人物シミュレーションのコンフィグレーション

型	変数名	単位
int	simulate_people	-
int	simulate_skip	-

simulate_people が非ゼロのときは人物のシミュレーションを行いますが，0 の場合は人物のシミュレーションを行わず，人が出現しない環境だけのシミュレーションになります．また，simulate_skip が 0 の場合はシミュレータを起動してからしばらく人物が出現していない時間がありますが，1 に設定すると起動直後から人物があらわれます．さらに，following_target_id が 0 以上で simulate_skip が 2 の場合には，追従対象人物が出現するところからシミュレーションを開始します．ただしこの場合にはシミュレータの起動に時間がかかるようになります．なお，simulate_people が 0 の場合は simulate_skip の設定は無効です．

2.4.6 robot_tread，robot_acceleration，robot_deceleration

表 8: ロボットの基本性能のコンフィグレーション

型	変数名	単位
double	robot_tread	m
double	robot_acceleration	m/s^2
double	robot_deceleration	m/s^2

これらのコンフィグレーションでロボットの基本的な性能を設定することができます．robot_tread はロボットのトレッド（車輪間距離），robot_acceleration はロボットの加速性能，そして robot_deceleration はロボットの減速性能 を決定するコンフィグレーションです．それぞれ想定するロボットに適した値を設定して下さい．

2.4.7 robot_init_x，robot_init_y，robot_init_heading

表 9: ロボット初期位置のコンフィグレーション

型	変数名	単位
double	robot_init_x	m
double	robot_init_y	m
double	robot_init_heading	degree

これらのコンフィグレーションで環境シミュレータ RTC 起動時のロボットの初期位置を設定することができます．robot_init_x で X 座標を，robot_init_y で Y 座標を，robot_init_heading で向きをそれぞれ設定します．このときの座標は大域座標系（図 3）で指定します．

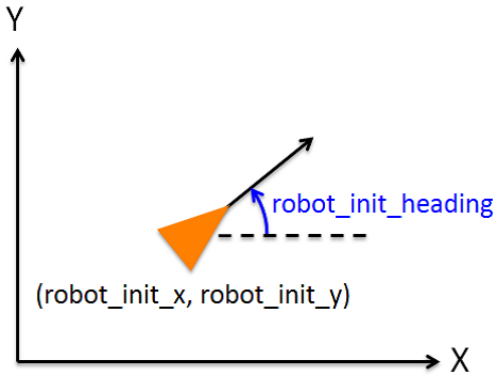

図 3: ロボット初期位置の座標系

2.4.8 robot_move_error_x，robot_move_error_y，robot_move_error_heading

表 10: ロボット走行誤差に関するコンフィグレーション

型	変数名	単位
double	robot_move_error_x	m
double	robot_move_error_y	m
double	robot_move_error_heading	degree

これらのコンフィグレーションでは，ロボットが移動したときにロボット制御命令に対して移動後の座標がどれだけずれるかを指定します．これらのコンフィグレーションの値はそれぞれ X 座標，Y 座標，向きに対する誤差の標準偏差を表しており，ロボットが 1m 移動するごとに移動後の位置姿勢がここで指定した値に応じてずれます．従って，これらの値を大きく設定するほど，ロボットは制御通り正確には進まなくなります．

2.4.9 range_start_position，range_end_position，range_data_num

表 11: レーザ距離データの計測範囲に関するコンフィグレーション

型	変数名	単位
double	range_start_position	degree
double	range_end_position	degree
int	range_data_num	-

range_start_position および range_end_position は距離センサで計測する開始角度と終了角度を決定します．つまり，この 2 つの値によってセンサの計測範囲が決まります．ここで角度の値は，図 4 に示すようにセンサ（ロボット）の右方向を角度の基準（0 °）とすることに注意して下さい．また，range_data_num は取得される距離データ数です．従って，角度分解能は

$\frac{range\_end\_position - range\_start\_position}{range\_data\_num}$ [degree] となります．

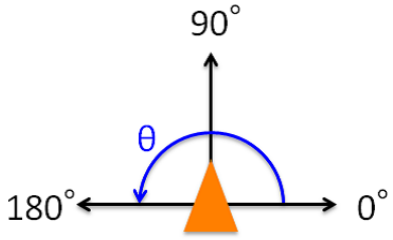

図 4: レーザ距離データの座標系

2.4.10 range_max_distance，range_far_code，range_std_deviation

表 12: レーザ距離データの計測可能距離に関するコンフィグレーション

型	変数名	単位
double	range_max_distance	m
long	range_far_code	-
double	range_std_deviation	m

range_max_distance は距離センサで計測できる最大距離です．これを超える距離は計測結果を得ることができないものとして，計測結果の距離値に range_far_code が格納されます．つまり，range_max_distance で表される距離よりも手前に障害物が存在しない場合に距離の値は range_far_code の値になります．

また，range_std_deviation は取得される距離データに付加される計測誤差の標準偏差です．

2.4.11 range_peason_radius

表 13: レーザ距離データに映る人物に関するコンフィグレーション

型	変数名	単位
double	range_peason_radius	m

レーザ距離データに人は円形に映ります．このときの人の半径を range_peason_radius で指定することができます．また，この値を 0 にすると距離データに人が現れないようにすることができます．

2.4.12 range_sensor_x，range_sensor_y，range_sensor_heading

表 14: レーザ距離データの取り付け位置に関するコンフィグレーション

型	変数名	単位
double	range_sensor_x	m
double	range_sensor_y	m
double	range_sensor_heading	degree

range_sensor_x および range_sensor_y は距離センサが取り付けられている位置の座標を，range_sensor_heading は角度を指定するためのコンフィグレーションです．これらの値は図 5 に示すようなロボット中心を基準とするロボットとの相対座標で指定します．これらのコンフィグレーションを設定することによって，ロボット中心からずれた位置に距離センサを設置した場合を想定した実験を行う事ができます．

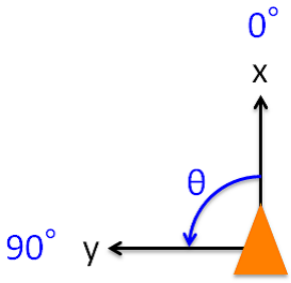

図 5: ロボットとの相対座標系

2.4.13 show_map_scale，display_range_data

表 15: 表示に関するコンフィグレーション

型	変数名	単位
double	show_map_scale	m/pixel
int	display_range_data	-

環境シミュレータ RTC を起動すると，シミュレーションの様子を表示するウィンドウが現れます．show_map_scale はこのときの画面に表示する環境（地図）のスケールを設定するためのものです．また display_range_data に 0 以外の値を指定すると，図 6 の様にシミュレータ上で再現されたレーザ距離データの計測した範囲を可視化して表示します．この図で青い線は各角度毎に計測した距離を表しています．

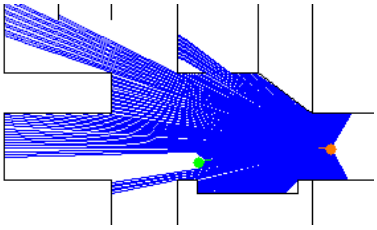

図 6: 可視化された距離データ

3　各データ型・インターフェースについて

この章では環境シミュレータコンポーネントで使用されている各データ型・インターフェースについて説明します.

3.1　SensorRTC::LaserRangeSensor::idl::MeasuredData

SensorRTC::LaserRangeSensor::idl::MeasuredData は株式会社セックが開発した北陽電機社 URG シリーズ用のコンポーネントで使用されているデータ型です．レーザ距離センサから距離データを取得するために用いられます.

- float startPosition: distance に最初に格納されているのデータの方向 [degree]
- float endPosition: distance に最後に格納されているのデータの方向 [degree]
- long scanInterval: スキャン間引き数
- long dataGroupingNumber: まとめるステップ数
- sequence⟨long⟩ distance: 各方向に対する距離データ [mm]
- float dataInterval: 各データの間の間隔 [degree]
- string sensorState: センサの状態 (例： ”NORMAL”, ”UPDATED”)

なお，startPosition および endPosition は図 7 に示すようにセンサ（ロボット）の右方向が角度の基準（0 °）となることに注意して下さい.

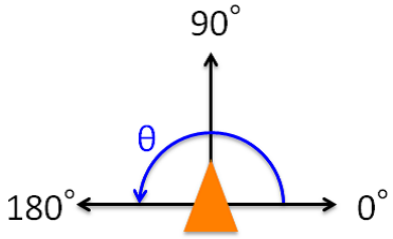

図 7: レーザ距離データの座標系

3.2　SensorRTC::LaserRangeSensor::idl::TimedMeasuredData

SensorRTC::LaserRangeSensor::idl::TimedMeasuredData は株式会社セックが開発した北陽電機社 URG シリーズ用のコンポーネントで使用されているデータ型です．レーザ距離センサから距離データを取得するために用いられます.

- MeasuredData data: 取得したデータ
- RTC::Time tm: タイムスタンプ

3.3 IIS::TimedPose2D

IIS::TimedPose2D はロボットの位置・姿勢を格納するデータ型です.

- Pose2D data: x（X 座標 [m]）, y（Y 座標 [m]）, heading（向き [radian]）を格納.
- error: 誤差分散を格納する．ただし，LocalMap コンポーネントではこの値は使用していない.
- id: 使用しない
- RTC::Time tm: タイムスタンプ

なお，ロボットの位置・姿勢には図 8 に示す座標系を用います.

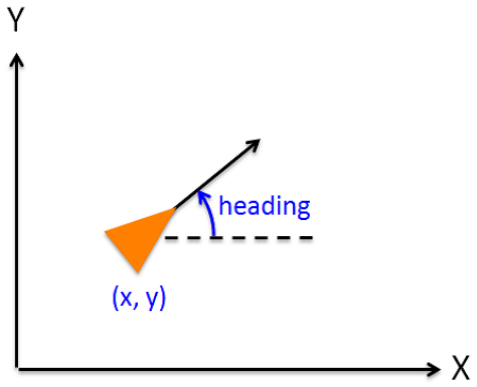

図 8: ロボット位置・姿勢の座標系

3.4 RTC::OGMapConfig

大きさやスケールといった地図の情報を格納するデータ型です.

- double xScale: X 軸方向の地図のスケール [m/cell]
- double yScale: Y 軸方向の地図のスケール [m/cell]
- double width: X 軸方向の地図の大きさ [cell]
- double height: Y 軸方向の地図の大きさ [cell]
- RTC::Pose2D origin: ロボット中心から見た cell(0,0) の絶対座標

3.5 MRFC::TimedRelativeOGMapData

地図を octed 型の系列で表現したデータ型です．ここで，octed 型は 8bit の符号付整数 (-128〜127) であり，各セル毎の障害物の存在確率を 0 から 100 の値で格納しています．また，そのセルが未観測の場合（未知領域の場合）は-1 が格納されます．セルの並びといった地図の仕様については図**??**も参照してください．

- RTC::OGMapConfig mapconfig: 地図の大きさやスケール
- RTC::OGMapCells cells: octed 型の系列，各セルの値を格納
- RTC::Pose2D pose: ロボット中心の絶対座標
- RTC::Time tm: タイムスタンプ

3.6 MRFC::TimedFloatRelativeOGMapData

地図を float 型の系列で表現したデータ型です．ここで，float 型は単精度浮動少数であり，各セル毎の障害物の存在確率を 0.0 から 1.0 の値で格納しています．また，そのセルが未観測の場合（未知領域の場合）は負の値が格納されます．セルの並びといった地図の仕様については図 9 も参照してください．

- RTC::OGMapConfig mapconfig: 地図の大きさやスケール
- RTC::OGMapFloatCells cells: float 型の系列，各セルの値を格納
- RTC::Pose2D pose: ロボット中心の絶対座標
- RTC::Time tm: タイムスタンプ

3.7 MRFC::RelativeMapService

MRFC::RelativeMapService は局所地図（ロボット座標系地図）を扱うためのインターフェースです．インターフェースに含まれるサービスは以下の通りです．

- RTC::OGMapConfig getRelativeOGMapConfig();
  地図の情報を取得.
- TimedRelativeOGMapData getRelativeOGMap();
  局所地図を取得.
- TimedFloatRelativeOGMapData getFloatRelativeOGMap();
  局所地図を取得.

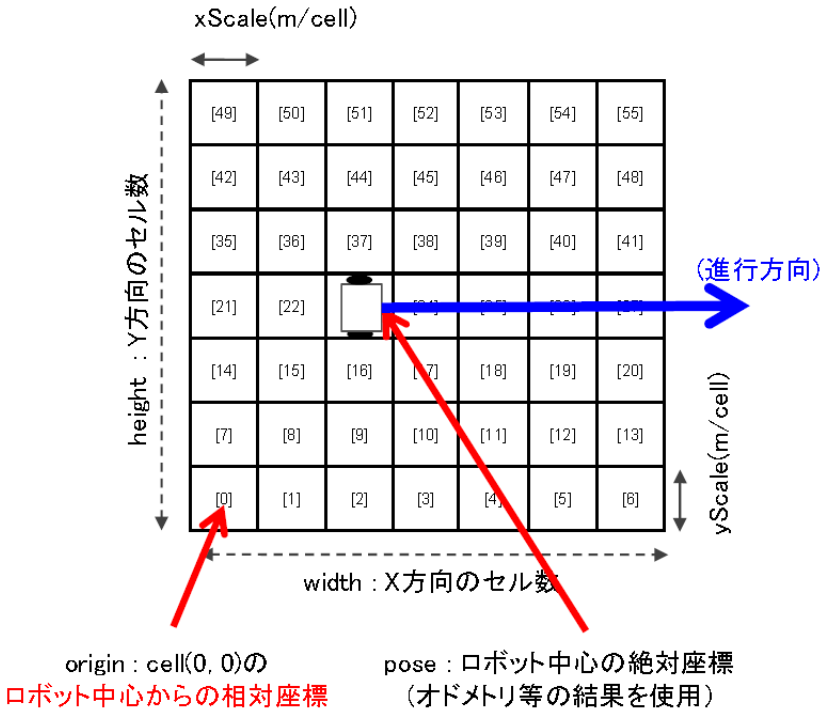

図 9: 局所地図の仕様

3.8 MRFC::TimedAbsoluteOGMapData

地図を octed 型の系列で表現したデータ型です．ここで，octed 型は 8bit の符号付整数 (-128〜127) であり，各セル毎の障害物の存在確率を 0 から 100 の値で格納しています．また，そのセルが未観測の場合（未知領域の場合）は-1 が格納されます．セルの並びといった地図の仕様については図**??**も参照してください．

- RTC::OGMapConfig mapconfig: 地図の大きさやスケール
- RTC::OGMapCells cells: octed 型の系列，各セルの値を格納
- RTC::Time tm: タイムスタンプ

3.9 MRFC::TimedFloatAbsoluteOGMapData

地図を float 型の系列で表現したデータ型です．ここで，float 型は単精度浮動少数であり，各セル毎の障害物の存在確率を 0.0 から 1.0 の値で格納しています．また，そのセルが未観測の場合（未知領域の場合）は負の値が格納されます．セルの並びといった地図の仕様については図 10 も参照してください．

- RTC::OGMapConfig mapconfig: 地図の大きさやスケール
- RTC::OGMapFloatCells cells: float 型の系列，各セルの値を格納
- RTC::Time tm: タイムスタンプ

3.10 MRFC::AbsoluteMapService

MRFC::AbsoluteMapService は大域地図（絶対座標系地図）を扱うためのインターフェースです．インターフェースに含まれるサービスは以下の通りです．

- RTC::OGMapConfig getAbsoluteOGMapConfig();
  地図全体の情報を取得．
- TimedAbsoluteOGMapData getAbsoluteOGMap(in double x, in double y, in unsigned long width, in unsigned long height);
  指定した範囲の地図を取得．セル (0,0) の絶対座標 x, y とセルサイズを指定する．
- TimedFloatAbsoluteOGMapData getFloatAbsoluteOGMap(in double x, in double y, in unsigned long width, in unsigned long height);
  指定した範囲の地図を取得．セル (0,0) の絶対座標 x, y とセルサイズを指定する．

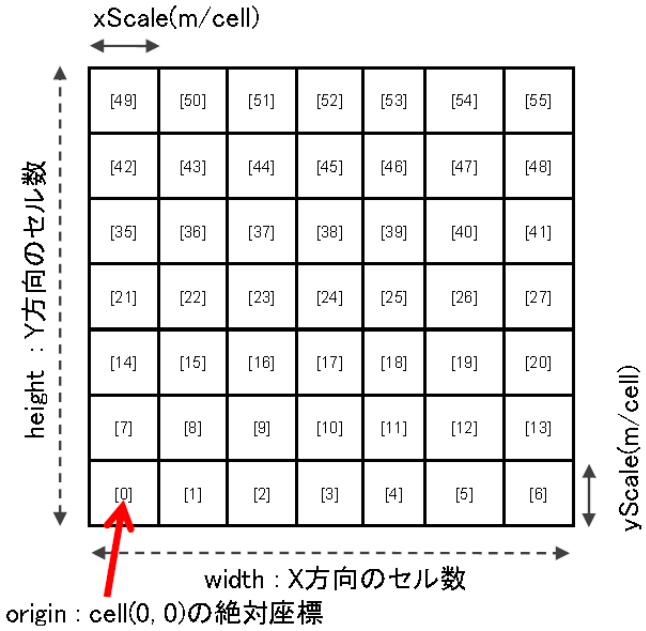

図 10: 大域地図の仕様

4　環境データについて

この章では，シミュレータが再現する環境について記述するデータの仕様を説明します．記述するデータは「壁（障害物）」,「目的地」,「入口」,「人の行動」および「人の初期行動」の５つに分類されます．

4.1　基本文法

要素は要素名（Walls, Destination, Entrance, Actions, FirstActions）の後ろに１つまたは大かっこ { } の中に複数記述します．各要素は値をコンマ「,」で区切り，セミコロン「;」で終わります．各要素については次節以降で説明します．

また,「#」の後ろに続く文字列はコメントとして処理されます．

4.2　壁（障害物）

壁の情報は要素名「Walls」の後ろに１つ，または { } の中に複数記述します．壁は 2 つの座標を結ぶ線分として表現し，次のように記述します．

(点 1 の x 座標 [m]), (点 1 の y 座標 [m]), (点 2 の x 座標 [m]), (点 2 の y 座標 [m]);

4.3　目的地

目的地の情報は要素名「Destination」の後ろに１つ，または { } の中に複数記述します．目的地は出口，地点，席，行列の 4 種類に分類され，各要素の始めに種類を表すアルファベットを記述します．2 番目には各目的地の番号を記述します．この番号は固有のものである必要はありません．同じ種類・同じ番号の目的地が複数ある場合は，その中の最も評価の高い目的地が選ばれます．

4.3.1　出口

出口の情報はアルファベット「X」を先頭に，出口を表す線分を指定する 2 つの座標からなり，以下の様に記述します．

X, (出口番号), (点 1 の x 座標 [m]), (点 1 の y 座標 [m]), (点 2 の x 座標 [m]), (点 2 の y 座標 [m]);

4.3.2　地点

地点の情報はアルファベット「P」を先頭に，座標と，その座標の何 m 以内に入れば到達したと判断するかを決める半径からなり，以下の様に記述します．

P, (地点番号), (x 座標 [m]), (y 座標 [m]), (半径 [m]);

4.3.3 席

席の情報はアルファベット「P」を先頭に，座標を記述します．地点との違いは，半径の指定が不要であることと，席は 1 人の人が占有するという 2 点です．つまり，他の人がすでに存在する席が目的地に設定される事はありません．席は以下の様に記述します．

S, (席番号), (x 座標 [m]), (y 座標 [m]);

4.3.4 行列

行列の情報はアルファベット「Q」を先頭に，座標と，行列が形成される方向を表す dx，dy からなり，以下の様に記述します．

Q, (行列番号), (x 座標 [m]), (y 座標 [m]), (dx[m]), (dy[m]);

4.4 入口

入口の情報は要素名「Entrance」の後ろに１つ，または { } の中に複数記述します．入口は 2 つの座標を結ぶ線分として表現し，線分上から人が出現します．また，何秒おきに人が出現するかという情報も入口に含まれます．入口は次のように記述します．

(入口番号), (点 1 の x 座標 [m]), (点 1 の y 座標 [m]), (点 2 の x 座標 [m]), (点 2 の y 座標 [m]), (平均出現間隔 [秒]), (出現間隔の標準偏差 [秒]);

4.5 人の行動

人の行動の情報は要素名「Actions」の後ろに１つ，または { } の中に複数記述します．行動とは，特定の目的地への移動とそこで停止する時間の事を指します．よって，行動の情報にはその行動の番号，移動先の目的地と停止時間，さらに行動を終えた後遷移する行動の番号を含みます．なお，行動の番号は固有の値でなければなりません．人の行動は次の様に記述します．

(行動番号), (目的地の種類), (目的地の番号), (次の行動), (平均停止時間 [秒]), (停止時間の標準偏差 [秒]);

4.6 人の初期行動

人の初期行動は要素名「FirstActions」の後ろに並べて，次の様に記述します．

FirstActions (0 番の入口から出現した人の最初の行動の番号), (1 番の入口から出現した人の最初の行動の番号), ...;

5　コンポーネントの実行手順

この章では，使用するための手順について説明します．

5.1　各プログラムの起動

まず初めにネームサーバ，RT System Editor および各コンポーネントを起動する必要があります．ネームサーバは，

スタート＞すべてのプログラム＞ OpenRTM-aist ＞ C++＞ tools ＞ Start Naming Service

を選択することで起動することができます．RT System Editor も同様に，

スタート＞すべてのプログラム＞ OpenRTM-aist ＞ C++＞ tools ＞ RT System Editor

を選択することで起動することができます．
次に各 RT コンポーネントを起動します．展開したフォルダの下にある『EnvironmentSimulator-Comp.exe』を実行して下さい．また，環境シミュレータ RTC と接続して用いる各コンポーネントを起動して下さい．

5.2　RT System Editor 上でのコンポーネントの接続

RT System Editor の起動とコンポーネントの接続は次のような手順で行うことができます．

1. eclipse を起動し，パースペクティブで RT System Editor を選択する．
2. 図 11 の赤い丸で囲んだアイコン『ネームサーバを追加』を選択する．
3. 図 11 のように『ネームサーバに接続』の Adress Port に『localhost』と入力して OK を選択する．
4. NameServiceView に起動したモジュールが表示されていることを確認する．
5. ファイル＞ Open New System Editor を選択する．
6. NameServiceView 上の RTC を選択して，System Editor 上にドラッグしてモジュールのアイコンを表示させる．
7. 環境シミュレータ RTC やその他の RTC を接続する．

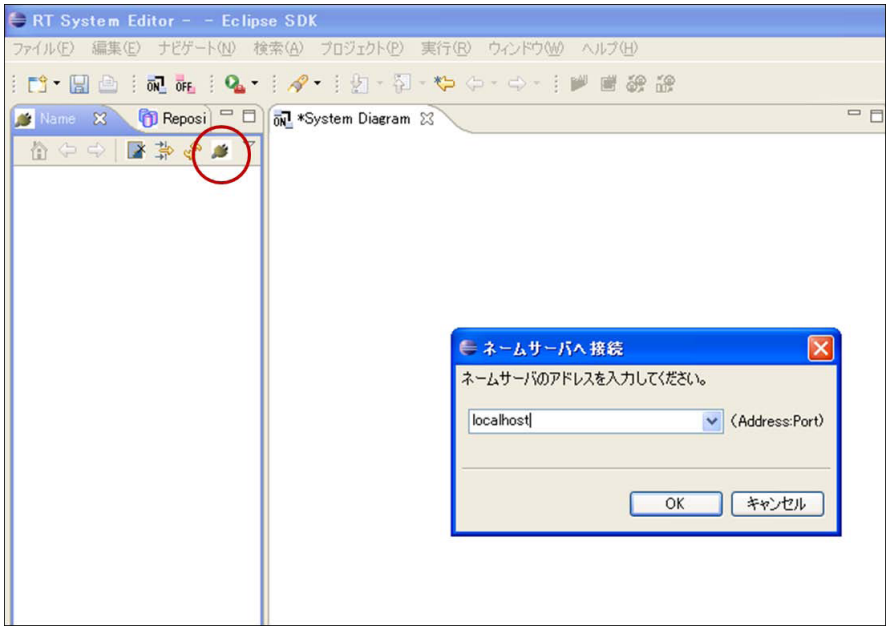

図 11: RT System Editor の画面

5.3 コンフィギュレーションの設定

RT System Editor 上で環境シミュレータ RTC を選択すると，ConfigurationView に図 12 のように表示されます．ここで必要に応じてコンフィギュレーションの Value を設定し,『適用』ボタンを押すことで値を変更できます.

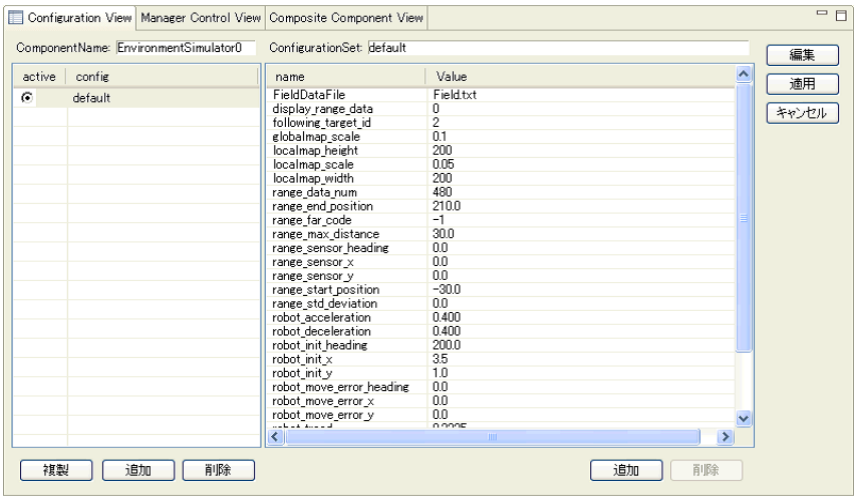

図 12: ConfigurationView

5.4　コンポーネントの実行と動作の確認

使用する RT コンポーネントの接続が完了し準備が整えば，全てのコンポーネントをアクティベートすることでシミュレータを実行することができます．実行すると図 13 のようにシミュレーションの様子が表示されます．このウィンドウにではロボットが橙色，人が緑色で表示されます．

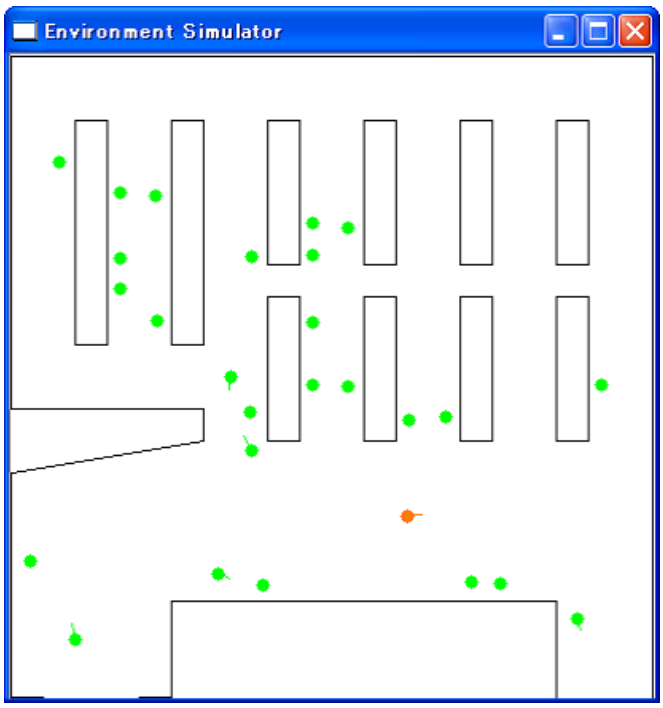

図 13: シミュレーションの様子

6　連絡先

豊橋技術科学大学 行動知能システム学研究室

〒 441-8580
愛知県豊橋市天伯町雲雀ヶ丘 1-1
豊橋技術科学大学 情報・知能工学系
行動知能システム学研究室
TEL: 0532-44-6826
URL: http://www.aisl.cs.tut.ac.jp/

不明な点がある場合は rtc@aisl.cs.tut.ac.jp まで連絡をお願いします.