自己位置推定コンポーネント
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 23 年 10 月 27 日

目 次

1 初めに **2**
1.1 ファイルの展開 2
1.2 開発・動作環境 2

2 各コンポーネントについて **2**
2.1 自己位置推定（Localization）コンポーネント 3
2.2 SimpleGlobalMapLoader コンポーネント 4

3 各データ型・インターフェースについて **5**
3.1 SensorRTC::LaserRangeSensor::idl::MeasuredData 5
3.2 SensorRTC::LaserRangeSensor::idl::TimedMeasuredData 5
3.3 IIS::TimedPose2D 6
3.4 MRFC::TimedEstimatedPose2D 6
3.5 TUT::ImageData 7
3.6 TUT::TimedImageData 7
3.7 TUT::StereoData 7
3.8 TUT::TimedStereoData 7
3.9 RTC::OGMapConfig 8
3.10 MRFC::TimedAbsoluteOGMapData 8
3.11 MRFC::TimedFloatAbsoluteOGMapData 8
3.12 MRFC::AbsoluteMapService 9

4 コンポーネントの実行手順 **10**
4.1 各プログラムの起動 10
4.2 RT System Editor 上でのコンポーネントの接続 10
4.3 コンフィギュレーションの設定 12
4.4 コンポーネントの実行と動作の確認 12

5 連絡先 **13**

1　初めに

このドキュメントでは，大域地図上でのロボット自己位置を推定する Localization コンポーネントおよび大域地図をファイルから読み込むための SimpleGlobalMapLoader コンポーネントについて解説し，その使い方を説明します.

1.1　ファイルの展開

ファイルの中身は図 1 のようになっています.

- Localization
  ロボット自己位置を推定するコンポーネントの実行ファイルを含むフォルダ.
- SimpleGlobalMapLoader
  大域地図データをファイルから読み込み出力するコンポーネントの実行ファイルを含むフォルダ.
- IIS2.idl，MRFC.idl および StereoCameraService.idl
  このコンポーネント群独自のデータ型・サービスを定義した IDL ファイル.

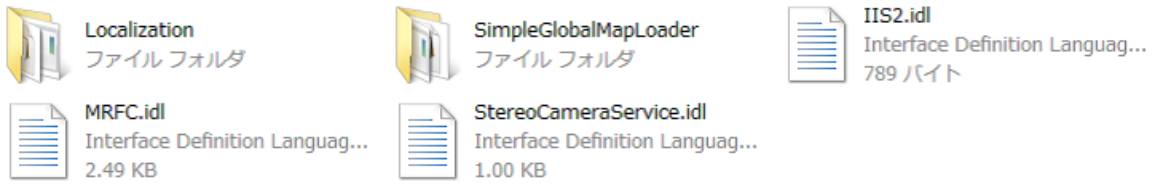

図 1: 解凍したフォルダの中身

1.2　開発・動作環境

各コンポーネント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，各コンポーネントは OpenCV 2.1 を使用しています．OpenCV については下記サイトを参照して下さい.

http://sourceforge.net/projects/opencvlibrary/

2　各コンポーネントについて

ここでは付属する各 RT コンポーネントについて説明します.

2.1 自己位置推定（Localization）コンポーネント

このコンポーネントは，データポートから入力された距離データとロボットの移動量を用いて大域地図上でのロボット位置を推定します．大域地図（障害物存在確率地図）はサービスポートからコンポーネント起動時に受け取ります．Localization コンポーネントが持つデータポートの一覧を表 1 に，サービスポートの一覧を表 2 に，コンフィギュレーションの一覧を表 3 に示します．

また，このコンポーネントは地図のスケールが 0.1[m/cell] の大域地図にしか対応していませんので，その他のスケールの地図が入力された場合には処理を行いません．

表 1: 入出力データポート

Port Type	Data Type	Port Name	備考
In Port	SensorRTC::LaserRangeSensor::idl::TimedMeasuredData	RangeData	LRF からのデータ入力
In Port	IIS::TimedPose2D	RobotPose	ロボット移動量入力
In Port	TUT::TimedImageData	Image	画像入力
In Port	TUT::TimedStereoData	StereoData	ステレオ距離データ入力
Out Port	IIS::TimedPose2D	EstimatedPose	推定したロボット位置・姿勢の出力
Out Port	MRFC::TimedEstimatedPose2D	PosePair	推定したロボット位置・姿勢の出力

表 2: サービスポート

Port Type	Data Type	Port Name	備考
Service Consumer	MRFC::AbsoluteMapService	AbsoluteMapService	絶対座標系地図の取得

表 3: コンフィギュレーション

型	変数名	単位	備考
int	Particle_num	-	パーティクル数
int	debug_window	-	0 以外の値を指定するとデバッグ用のウィンドウを表示
int	USE_INIT_POSE	-	0 を指定するとパーティクルを均等に散布 0 以外の値を指定するとパーティクルを指定した位置を中心に散布
double	INIT_X	m	ロボット初期 X 座標（USE_INIT_POSE が非 0 のときに有効）
double	INIT_Y	m	ロボット初期 Y 座標（USE_INIT_POSE が非 0 のときに有効）
double	INIT_HEADING	radian	ロボット初期方向（USE_INIT_POSE が非 0 のときに有効）

2.2 SimpleGlobalMapLoader コンポーネント

このコンポーネントは大域地図データを画像ファイルから読み込み，それを基にサービスポートから地図を出力します．SimpleGlobalMapLoader コンポーネントが持つサービスポートの一覧を表 4 に，コンフィギュレーションの一覧を表 5 に示します．データポートはありません．

表 4: サービスポート

Port Type	Data Type	Port Name	備考
Service Provider	MRFC::AbsoluteMapService	AbsoluteMapService	絶対座標系地図の出力

表 5: コンフィギュレーション

型	変数名	単位	備考
string	MapData	-	読みこむ地図のファイル名を指定.
double	InputMapScale	m/pixel	読みこんだ地図画像のスケール.

読み込んだ大域地図画像データはまずグレースケール化され，そのときの画素値が 0 に近いほど障害物の存在確率が低く（ロボットが通れる），255 に近いほど高い（ロボットが通れない）ことを表すものとします．大域地図画像データの例を図 2 に示します．画像データは PNG といった OpenCV が読み込むことができる形式でなければなりません．

図 2: 大域地図画像の例

3　各データ型・インターフェースについて

ここでは Localization コンポーネントおよび SimpleGlobalMapLoader 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 3 に示すようにセンサの右方向が角度の基準（0 °）となることに注意して下さい.

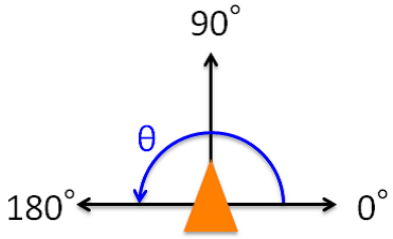

図 3: レーザ距離データの座標系

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，Localization 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 4 に示す座標系を用います.

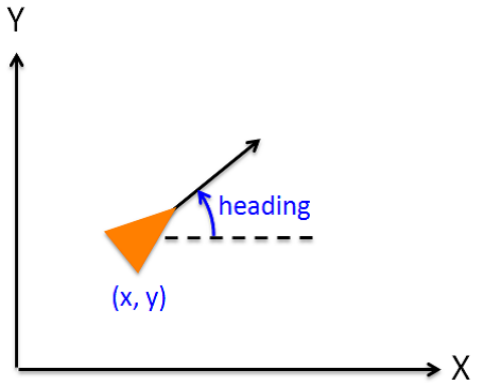

図 4: ロボット位置・姿勢の座標系

3.4 MRFC::TimedEstimatedPose2D

MRFC::TimedEstimatedPose2D は推定位置とその推定位置におけるオドメトリ値をペアで出力するためのデータ型です.

- Pose2D ododata: 推定位置におけるオドメトリ値
- Pose2D estdata: 推定された自己位置
- error: 誤差分散を格納する．ただし，Localization 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

3.5 TUT::ImageData

一枚の画像データを格納するデータ型です．このデータ型は OpenCV の IplImage 型を基にしていますので，各変数については OpenCV のマニュアル等が参考になります．

- long nChannels: チャンネル数（1,2,3,4 のどれか）
- long depth: 1 画素あたりのビット数
- long origin: 画像データの原点（基準）0:左上原点（デフォルト），1:左下原点
- long width: 画像の横方向の画素数
- long height: 画像の縦方向の画素数
- long imageSize: 画像データのサイズ（バイト数）
- sequence⟨char⟩ imgData: 画素値の系列
- long widthStep: 画像の横一行分のバイト数（画素数ではない）

3.6 TUT::TimedImageData

タイムスタンプ付きの画像データです．

- RTC::Time tm: タイムスタンプ
- ImageData data: 画像データ

3.7 TUT::StereoData

ステレオ距離画像の一画素分の距離データを格納するデータ型です．

- double x: X 座標
- double y: Y 座標
- double z: Z 座標，距離が取得できない場合には負の値を格納
- sequence⟨short⟩ dmy: 不使用

3.8 TUT::TimedStereoData

タイムスタンプ付きのステレオ距離画像です．data には最も左上の画素に対応するデータを先頭に，画像の画素データと同じ順番で値が格納されています．

- RTC::Time tm: タイムスタンプ
- long width: 画像サイズ（横）
- long height: 画像サイズ（縦）
- sequence⟨StereoData⟩ data: 距離データの系列

3.9 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．なお，大域地図を扱う場合の origin の姿勢情報には常に 0 が格納されます．

- 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

3.10 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 5 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

3.11 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 5 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

3.12 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．

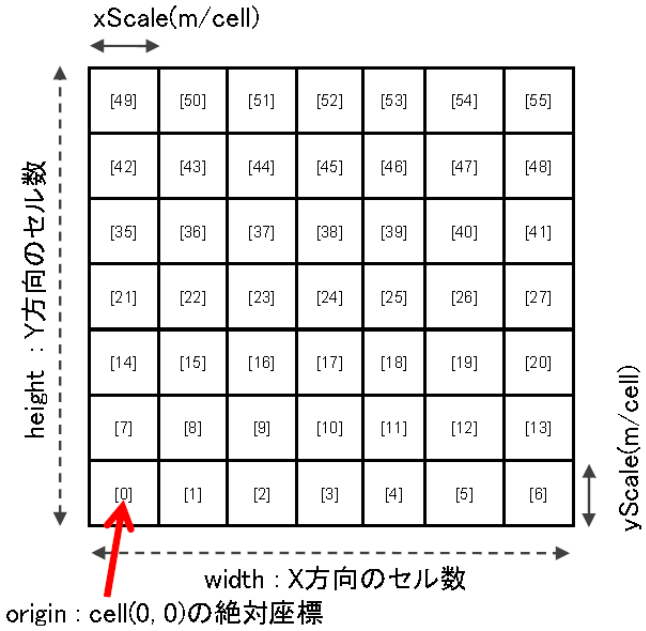

図 5: 大域地図の仕様

4　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．

4.1　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『LocalizationComp.exe』，『SimpleGlobalMapLoaderComp.exe』を実行して下さい．また，Localization コンポーネントにデータを入力するために用いる各コンポーネントを起動して下さい．ここでは例として URGDataFlowComp（北陽電機社製レーザセンサのデータを取得するコンポーネント），MobileRobotController コンポーネント（Mobile Robot 社ロボット用制御コンポーネント）および Bumblebee2Module コンポーネント（Point Grey 社製ステレオカメラのデータを取得するコンポーネント）を用います．

4.2　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 6 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 6 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 上のモジュールを選択して，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. 図 7 のようにモジュールを接続する．また，図 8 のようにステレオカメラを用いない構成でも自己位置を推定する事ができる．

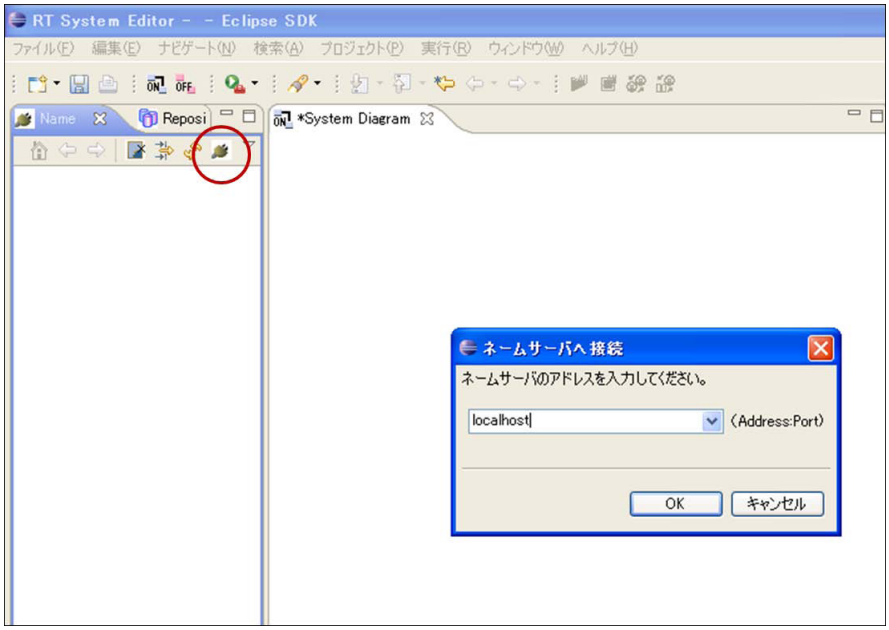

図 6: RT System Editor の画面

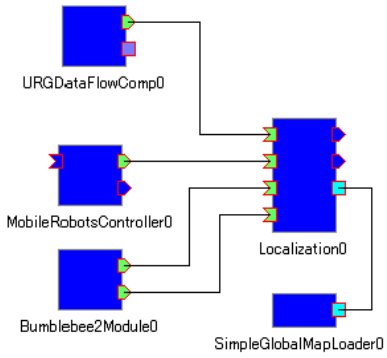

図 7: 各 RT コンポーネントの接続

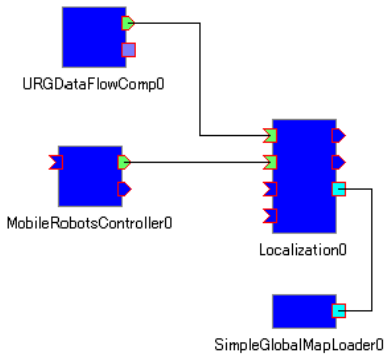

図 8: ステレオカメラを使用しない構成での各 RT コンポーネントの接続

4.3 コンフィギュレーションの設定

RT System Editor 上で Localization コンポーネントを選択すると，ConfigurationView に図 9 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．

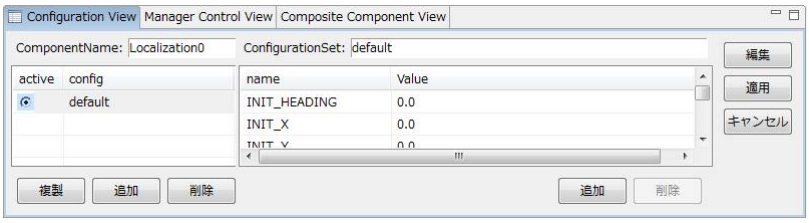

図 9: ConfigurationView

4.4 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自己位置推定を開始することができます．Localization コンポーネントのコンフィグレーションで debug_window に 0 以外の値を指定すると，図 10 の様なデバッグ用のウィンドウが表示されます．この画面では左上にカメラ画像，左下にステレオ距離画像，右側に地図が表示されます．ここで，地図は X 座標が上，Y 座標が左の座標系で表示されます．入力した地図画像とは表示される向きが異なる事に注意して下さい．また，カメラ画像の抽出された特頂点から対応する地図の箇所に向かって白い矢印が表示されますが，これは最も尤度の高いパーティクルの示す位置に表示されており，自己位置が正しく推定されていなければ正しい位置に矢印が表示されません．

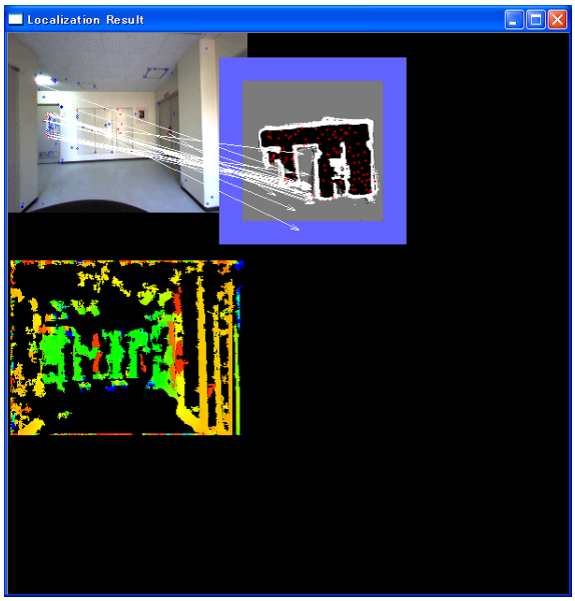

図 10: デバッグ用ウィンドウの表示

なお，今回例としてあげた構成ではロボットを移動させることができませんが，Localization コンポーネントは本来ロボットが移動しながら位置を推定する事を前提としたコンポーネントです．MobileRobotController コンポーネントに移動命令を入力することで動き回りながらそのロボット自己位置を推定する事ができます．移動しながら自己位置を推定した結果は図 11 の様に表示されます．地図上の赤い点が各パーティクルの位置を示しています．

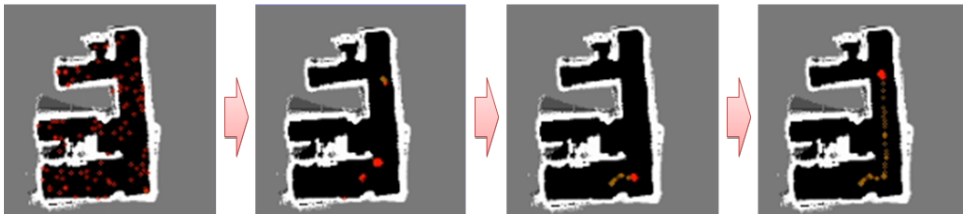

図 11: 自己位置推定結果の例

5 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．