次世代ロボット知能化技術開発プロジェクト

ロボット知能ソフトウェア再利用性向上技術の開発

機能仕様書

オープンソース移動知能モジュール群

障害物検知・衝突回避モジュール編

Ｖｅｒ．１.０

２０１１年6月３０日

ＲＴＣ再利用技術研究センター

改版履歴

Ver.	改版日	改版内容
1.0	2011/06/30	新規作成

目次

１．　はじめに

１．１．　本書の適用範囲

本書は、ロボット向けミドルウェア OpenRTM 上で、LRF によるスキャンデータを用いて、障害物検知、衝突判定を行い、障害物への回避行動を行う知能モジュールについて記述した文書である。

本書は「次世代ロボット知能化技術開発プロジェクト」における移動知能モジュール群として構成し、動作確認したものである。本構成以外の利用における有用性、汎用性について保証するものではない。

１．２．　関連文書

本書は以下に示す移動知能モジュール関連文書の一部である。

表 1-1 関連文書

No.	文書名	備考
1	機能仕様書 自己位置姿勢推定モジュール	
2	機能仕様書 走行系	
3	機能仕様書 経路計画・軌道追従	
4	機能仕様書 障害物検知・衝突回避	本書
5	機能仕様書 オペレータ操作	

１．３．　本書を読むにあたって

本書は RT ミドルウェア(以下 RTM)、RT コンポーネント(以下 RTC)を用いたロボットシステム開発者を対象に記述されており、RTM、RTC や関連ツールに関する一般的な知識を持つことを前提とする。RT ミドルウェア、RTC については下記を参照のこと。

OpenRTM-aist Official Website：

http://www.openrtm.org/

２．　機能仕様

本知能モジュールについての機能仕様、構成を以下に記述する。

２．１．　機能概要

本知能モジュールは対向二輪型の移動ロボットの走行時に、障害物を検知し、状況に合った障害物回避動作を実現する。本知能モジュールでは、LRF キャプチャ、障害物検知、障害物モニター、衝突判定、回避行動が機能として提供されている。

２．２．　モジュール構成

本知能モジュールは以下の構成で動作する。移動知能モジュール全体での位置付けは、移動知能モジュール群全体図を参照のこと。

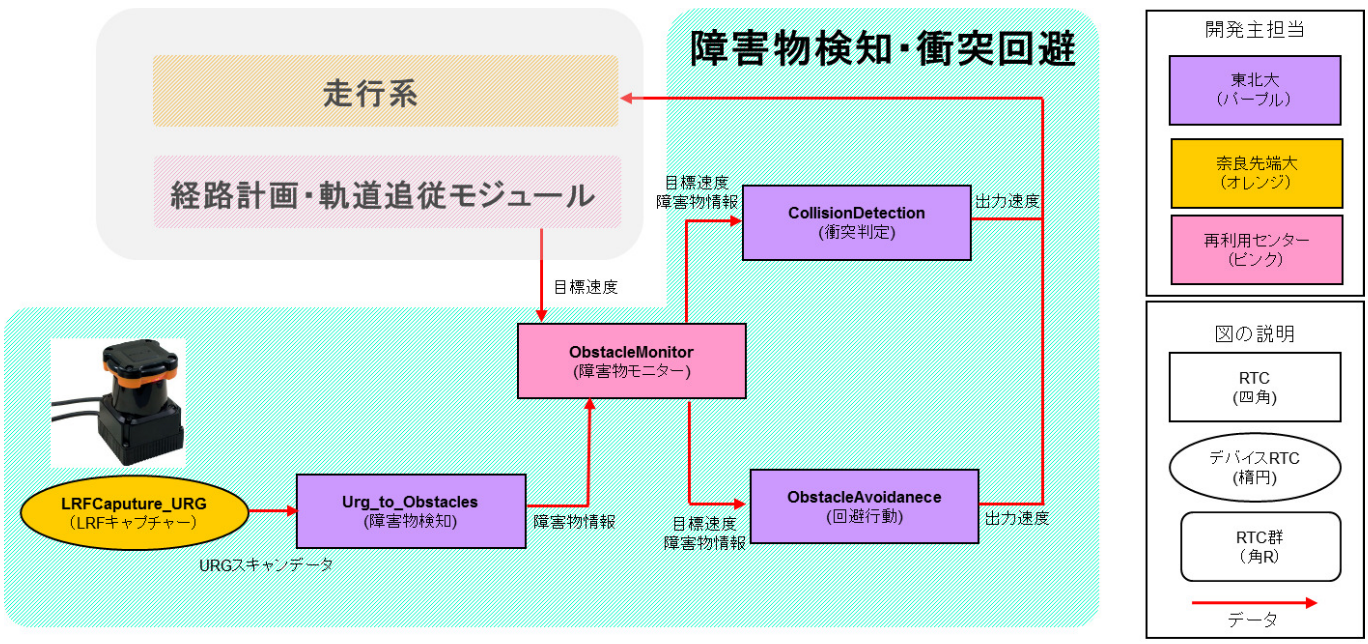

図　2-1 障害物検知・衝突回避モジュール構成図

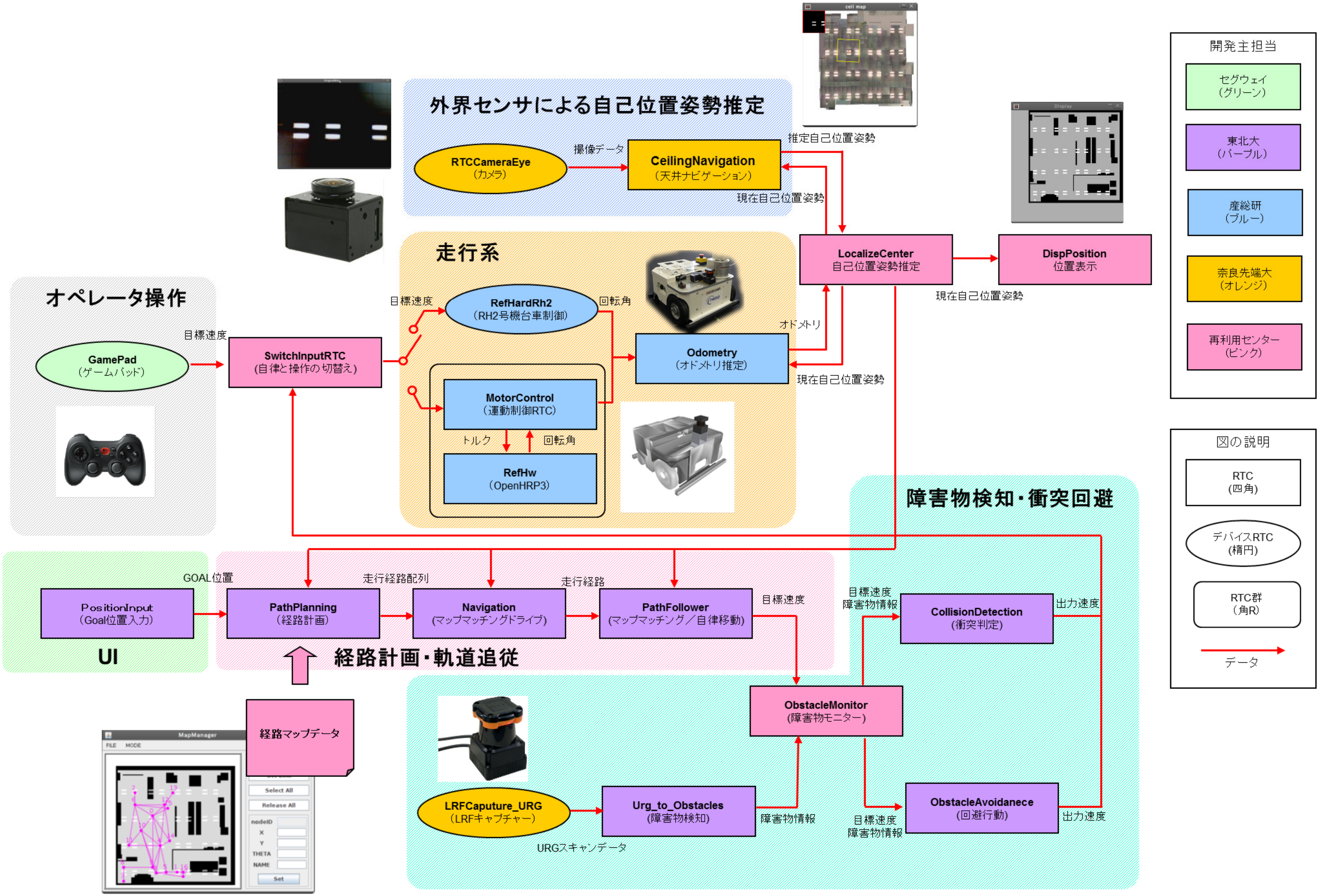

図 2-2 移動知能モジュール群全体図

３．　RTC 仕様

３．１．　LRFCapture_URG（LRF キャプチャコンポーネント）

３．１．１．　機能概要

本 RTC は、HOKUYO URG ライブラリ 0.8.12 を介して、以下のような北陽電機製 URG から距離データを取得するコンポーネントである。

図 ３-1 北陽電機製 URG　UTM-30LX

３．１．２．　動作環境

本 RTC の動作環境（動作 OS、RT ミドルウェア、開発環境など）について記述する。

動作 OS	Linux（Ubuntu10.04 LTS）
開発言語	C/C++
コンパイラ	G++コンパイラ 4.4.3
RT ミドルウェア／バージョン	OpenRTM-aist-1.0.0-RELEASE
依存パッケージ	HOKUYO 製 URG サンプルライブラリ 0.8.12 C++ Library boost(>=1.35) SDL(>=1.2.10) library

また、URG（UTM-30LX）のファームウェアは最新のもの（本体ソフト 1.18.01）を使用した。詳細については以下の URL を参照のこと。

HOKUYO URG シリーズ用ダウンロードページ
http://www.hokuyo-aut.co.jp/02sensor/07scanner/download/index.html

３．１．３．　動作条件

本 RTC のコンフィグレーションファイル（rtc.conf）の設定の内、ユーザが注意すべき項目を以下に明記する。

実行型	PERIODIC
実行周期	40[Hz]

３．１．４．　ポート情報

A)　データポート（OutPort）

名称	型	データ長	説明
RangeData	TimedOctetSeq	info.total_points	距離データ列［mm］

B)　サービスポート（Provider）

サービス名	インターフェース名	説明
LRF	LRFInfo	計測設定情報

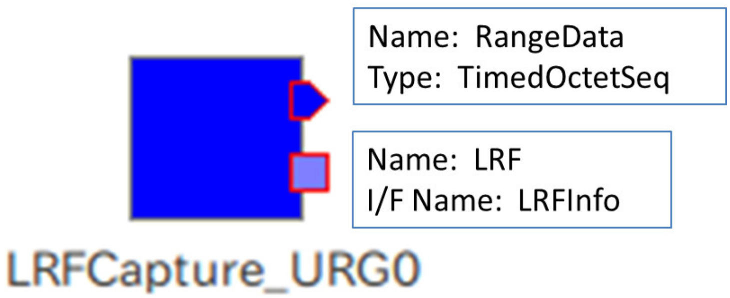

図　３-2　LRFCapture_URG 単体モジュール構成図

３．１．５．　コンフィグレーション

1）device_port

URG と通信するシリアルポートデバイス。デフォルトは auto。

auto にすると自動で接続ポートを検索する。

直接指定する場合は、/dev/ttyACM0 などを入力する。

2）start_deg

計測範囲の始点位置。 デフォルトは-90。 [deg]

3 時方向を 0[deg]する反時計回り。

設定できない値が指定された場合、計測可能範囲に自動で調節する。

実際に設定された値はサービスポートの関数 getOffsetFOV() より取得できる。

3) end_deg

計測範囲の終点位置。デフォルトは 270[deg] 。3 時方向を 0[deg]する反時計回り。

設定できない値が指定された場合、計測可能範囲に自動で調節する。

実際に設定された値はサービスポートの関数より計算できる。

(offset + total_points * resolution)

デフォルト値の他に複数コンフィギュレーション値が config_lrf.conf に記述されている。

A) mode 0 : デフォルト値

名称	型	値	説明
device_port	stirng	auto	URG と通信するシリアルポートデバイス
start_deg	double	-90	計測範囲の始点位置
end_deg	double	270	計測範囲の終点位置

B) mode 1

名称	型	値	説明
device_port	stirng	auto	URG と通信するシリアルポートデバイス
start_deg	double	0	計測範囲の始点位置
end_deg	double	180	計測範囲の終点位置

C) mode 2

名称	型	値	説明
device_port	stirng	auto	URG と通信するシリアルポートデバイス
start_deg	double	-45	計測範囲の始点位置
end_deg	double	225	計測範囲の終点位置

３．１．６． 入出力データフォーマット

以下にデータ型および詳細について示す。

A) 出力：RangeData

型：TimedOctetSeq

```
struct TimedOctetSeq {
    Time tm;
    sequence<octet> data;
};
```

要素	説明
data	距離データ列 [mm]

３．１．７． サービスインターフェース仕様

各サービスインターフェース仕様を本章に記述する。

３．１．７．１． インターフェース名

A) 基本情報

インスタンス名	m_LRFInfoPort
変数名	m_LRF
IDL ファイル	LRFInfo.idl
インターフェース型	LRFInfoSVC_impl

B) サービス関数一覧

No.	関数名	説明
1	getTotalScanPoints()	計測される総点数［points］
2	getResolution()	計測解像度［rad/scan］
3	getOffsetFOV()	計測範囲のオフセット［rad］３時方向を 0rad とする反時計回り.

C) サービス関数詳細

関数名	getTotalScanPoints()	
引数	なし	
戻り値	型	説明
	CORBA::Short	計測総点数
説明	計測される LRF 総点数［points]を返す。	

関数名	getResolution()	
引数	なし	
戻り値	型	説明
	CORBA::Double	計測解像度
説明	計測された LRF 解像度［rad/scan]を返す。	

関数名	getOffsetFOV()	
引数	なし	
戻り値	型	説明
	CORBA::Double	計測範囲オフセット
説明	計測範囲のオフセット[rad]を返す．３時方向を 0rad とする反時計回り	

３．１．８． 設定ファイル

本 RTC は、以下のライブラリを使用している。

１）HOKUYO 製 URG サンプルライブラリ 0.8.12 （URG 製品資料を参照）

２）C++ Library required boost(>=1.35) and SDL(>=1.2.10) library.

(a)依存ライブラリのインストール

```
$ sudo aptitude update
$ sudo aptitude install libsdl1.2-dev libsdl-net1.2-dev
$ unzip urg-0.8.12.zip
$ cd urg-0.8.12
$ ./configure
$ make
$ sudo make install
```

(b)PATH の設定

Makefile 内の URG_FLGS、 URG_LIBS を URG ライブラリのインストール PATH に合わせて修正する。

３．２．　Urg_to_Obstalces（障害物検知コンポーネント）

３．２．１．　機能概要

本 RTC は測域センサから得られた情報を、障害物情報に変換するコンポーネントである。障害物検知コンポーネントは、衝突判定コンポーネント、回避行動コンポーネントの実行に必要なモジュールである。

本 RTC を利用し障害物回避動作を実現するためには、測域センサ情報を取得するモジュール、回避経路を計画するモジュールが別途必要である。

３．２．２．　動作環境

本 RTC の動作環境（動作 OS、RT ミドルウェア、開発環境など）について記述する。

動作 OS	Linux（Ubuntu10.04 LTS）
開発言語	C/C++
RT ミドルウェア／バージョン	OpenRTM-aist-1.0.0-RELEASE
依存パッケージ	なし

３．２．３．　ポート情報

本 RTC のポート情報を記述する。

A）データポート（InPort）

名称	型	データ長	説明
urg	TimedLongSeq	m_urg.data.length()	URG のデータ

B）データポート（OutPort）

名称	型	データ長	説明
obstacle	TimedObstacles	障害物数	障害物の情報

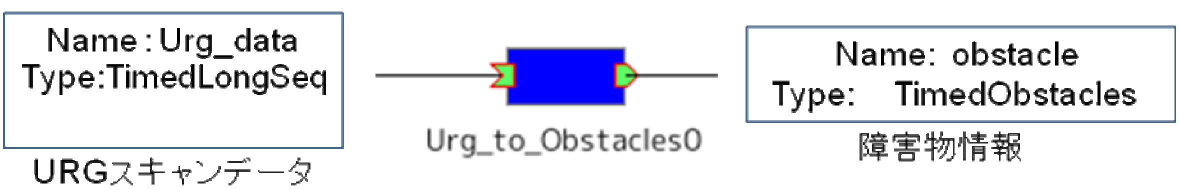

図　3-3 Urg_to_Obstacles 単体モジュール構成図

３．２．４．　コンフィグレーション

コンフィグレーション設定について記載する。

複数のコンフィグレーションセットを保有する場合は各設定値も記載する。

名称	型	デフォルト値	説明
obstacle_max_radius	double	0.05	障害物の最大半径[m]
max_dist	double	3.0	検出する最大距離[m]
urg_res	double	0.25	URG の角度分解能[deg]
urg_anglefront	int	540	URG のステップの中心値[step]
urg_anglemin	int	0	URG のステップの最小値[step]
urg_pos_x	double	0.0	URG の取り付け位置［m]
urg_pos_y	double	0.0	URG の取り付け位置［m]
urg_pos_theta	double	0.0	URG の取り付け角[rad]

３．２．５．　入出力データフォーマット

以下に動作概略およびデータ型詳細について示す。

1)　動作

測域センサの情報から、障害物情報を抽出する。

障害物情報として変換する際、次の処理を行う。

- 適度な大きさに間引く
- ロボット中心座標系に変換
- 障害物サイズ情報を付加

2)　データ型詳細

A）入力：urg

型：TimedLongSeq

```
struct TimedLongSeq {
    Time tm;
    sequence<long> data;
};
```

要素	説明
data[id]	URG からの距離データ

※id は URG データ数

B）出力：obstacle
型：TimedObstacles

```
struct TimedObstacles{
    Time tm;
    sequence<TimedObstacle> obstacles;
};
```

要素	説明
obstacles[id].x	障害物 X 位置
obstacles[id].y	障害物 Y 位置
obstacles[id].r	障害物半径[m]

※id は障害物数

３．２．１．　設定ファイル

本モジュールでは、特殊なドライバ、ライブラリを使用しておらず、使用するにあたって、OpenRTM-aist-1.0.0-RELEASE・RTSystemEditor の標準的な設定で問題はなく、新たにドライバ、ライブラリなどのインストールは必要ない。

３．３． ObstacleMonitor（障害物モニタコンポーネント）

３．３．１． 機能概要

本 RTC は障害物情報に変換された測域センサからの情報を利用し、衝突判定、回避行動モジュールへ障害物情報を送出するコンポーネントである

また、本 RTC は経路計画・起動追従モジュールからの目標速度情報を統合し、衝突判定・障害物回避モジュールへの目標速度を送出する機能も有している。

３．３．２． 動作環境

本 RTC の動作環境（動作 OS、RT ミドルウェア、開発環境など）について記述する。

動作 OS	Linux（Ubuntu10.04 LTS）
開発言語	C/C++
RT ミドルウェア／バージョン	OpenRTM-aist-1.0.0-RELEASE
依存パッケージ	なし

３．３．３．　ポート情報

本 RTC のポート情報を記述する。

A) データポート（InPort）

名称	型	説明
in_vel	IIS:: TimedVelocity2D	目標車体速度
obstacles	TimedObstacles	障害物の情報

B) データポート（OutPort）

名称	型	説明
out_vel_stp	IIS::TimedVelocity2D	目標車体速度(一般停止側)
obstacles_stp	TimedObstacles	障害物情報(一般停止側)
out_vel_avd	IIS::TimedVelocity2D	目標車体速度(回避側)
obstacles_avd	TimedObstacles	障害物情報(回避側)
state	IIS::TimedState	遷移状態

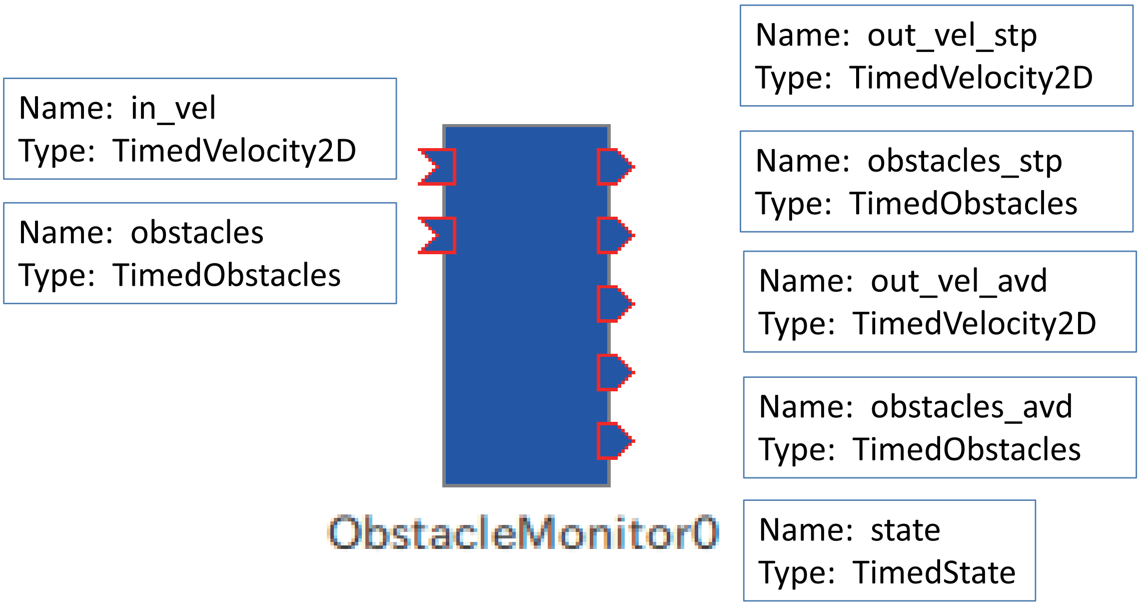

図　３-4 ObstacleMonitor 単体モジュール構成図

３．３．４．　コンフィグレーション

コンフィグレーション設定について記載する。

名称	型	デフォルト値	説明
body_radius	double	0.1	ロボットの半径[m]
speed_weight	double	0.3	回避時の速度に関する重み係数
turn_weight	double	0.3	回避時の角速度に関する重み係数
length_weight	double	0.3	走行距離に関する重み計数
interval_length	double	0.3	衝突判定処理を行う間隔[m]
space	double	0.5	車間距離
acceleration	double	0.1	減速度[m/s^2]
point_thread_1	double	400.0	未使用
point_thread_2	double	550.0	未使用
waittime	double	5.0	待機時間[sec]

３．３．５．　入出力データフォーマット

以下にデータ型詳細について示す。移動サブワーキンググループの共通インタフェース仕様については、「移動 SWG 共通 IF 案 101008.pdf」を参照のこと。

A）入力：in_vel
型：IIS::TimedVelocity2D

```
struct TimedVelocity2D {
  RTC::Time tm;
  sequence<long> id;
  RTC::Velocity2D data;
  sequence<double> error;
};
```

要素	説明
data.vx	移動速度[m/s]
data.va	回転速度[rad/s]

B) 入力：obstacle

型：TimedObstacles

```
struct TimedObstacles{
    Time tm;
    sequence<TimedObstacle> obstacles;
};
```

要素	説明
obstacles[id].x	障害物 X 位置
obstacles[id].y	障害物 Y 位置
obstacles[id].r	障害物半径[m]

※id は障害物数

C) 出力：out_vel_stp、out_vel_avd

型：IIS::TimedVelocity2D

```
struct TimedVelocity2D {
    RTC::Time tm;
    sequence<long> id;
    RTC::Velocity2D data;
    sequence<double> error;
};
```

要素	説明
data.vx	移動速度[m/s]
data.va	回転速度[rad/s]

D) 出力：obstacles_stp、obstacles_avd

型：TimedObstacles

```
struct TimedObstacles{
    Time tm;
    sequence<TimedObstacle> obstacles;
};
```

要素	説明
obstacles[id].x	障害物 X 位置
obstacles[id].y	障害物 Y 位置
obstacles[id].r	障害物半径[m]

※id は障害物数

E）出力：state
型：IIS:: TimedState

```
struct TimedState {
  RTC::Time tm;
  long id;
  boolean isStop;
};
```

要素	説明

id	遷移状態
	enum {
	INITIAL = 0,
	NO_OBSTACLE,
	STOP,
	WAITING,
	AVOID,
	WARNING,
	RUNNING
	};

３．３．６．　設定ファイル

本モジュールでは、特殊なドライバ、ライブラリを使用しておらず、使用するにあたって、OpenRTM-aist-1.0.0-RELEASE・RTSystemEditor の標準的な設定で問題はなく、新たにドライバ、ライブラリなどのインストールは必要ない。

３．４． CollisionDetection（衝突判定コンポーネント）

３．４．１． 機能概要

本 RTC は、目標車体速度にて障害物との衝突判定を行ない、安全に停止可能な速度を出力するコンポーネントである。

衝突判定コンポーネントは入力された車体速度に対して、障害物との衝突可能性を判定し、衝突する可能性がある場合、安全に停止可能な車体速度を計算し、衝突の検知には、車体の速度、角速度を考慮し、一定加速度にて減速した場合に衝突する可能性がある速度である場合、停止可能な速度に減速し、出力する。この際、速度の変更により経路が変更されないよう、角速度も同時に変更される。これにより、AGV 等経路が定められている環境において障害物との衝突を回避することができる。

本 RTC を利用するためには、目標車体速度を与えるモジュール、センサ情報から障害物情報を生成するモジュール、車体速度に応じて移動ロボットを制御するモジュールが別途必要である。

３．４．２． 動作環境

本 RTC の動作環境（動作 OS、RT ミドルウェア、開発環境など）について記述する。

動作 OS	Linux（Ubuntu10.04 LTS）
開発言語	C/C++
RT ミドルウェア／バージョン	OpenRTM-aist-1.0.0-RELEASE
依存パッケージ	なし

３．４．３．　ポート情報

本 RTC のポート情報を記述する。

A）データポート（InPort）

名称	型	データ長	説明
in_vel	IIS:: TimedVelocity2D	1	目標車体速度
obstacles	TimedObstacles	障害物数	障害物情報

B）データポート（OutPort）

名称	型	データ長	説明
out_vel	IIS:: TimedVelocity2D	1	停止車体速度

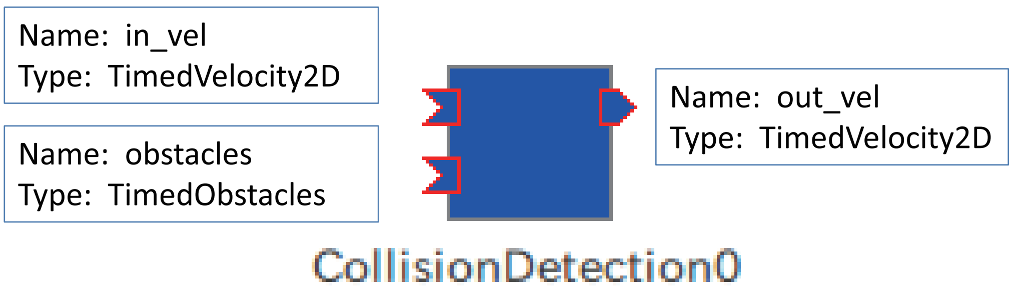

図　3-5 CollisionDetection 単体モジュール構成図

３．４．１．　コンフィグレーション

コンフィグレーション設定について記載する。

名称	型	デフォルト値	説明
body_radius	double	0.5	ロボットの半径[m]
acceleration	double	0.1	減速度[m/s²]
space	double	0.5	車間距離
interval_length	double	0.3	衝突判定処理を行う間隔[m]

３．４．２． 入出力データフォーマット

以下に動作およびデータ型詳細について示す。移動サブワーキンググループの共通インタフェース仕様については、「移動 SWG 共通 IF 案 101008.pdf」を参照のこと。

１）動作

入力された車体速度に対して、障害物と衝突する可能性があるか判定し、衝突する可能性がある場合、安全に停止可能な車体速度を計算し、衝突の検知には、車体の速度、角速度を考慮し、一定加速度にて減速した場合に衝突する可能性がある速度である場合、停止可能な速度に減速し、出力する。この際、速度の変更により経路が変更されないよう、角速度も同時に変更される。

２）データ型詳細

A）入力：in_vel

型：IIS::TimedVelocity2D

```
struct TimedVelocity2D {
    RTC::Time tm;
    sequence<long> id;
    RTC::Velocity2D data;
    sequence<double> error;
};
```

要素	説明
data.vx	移動速度[m/s]
data.va	回転速度[rad/s]

B）入力：obstacles

型：TimedObstacles

```
struct TimedObstacles{
    Time tm;
    sequence<TimedObstacle> obstacles;
};
```

要素	説明
obstacles[id].x	障害物 X 位置
obstacles[id].y	障害物 Y 位置

※id は障害物数

C）出力：out_vel

型：IIS::TimedVelocity2D

```
struct TimedVelocity2D {
    RTC::Time tm;
    sequence<long> id;
    RTC::Velocity2D data;
    sequence<double> error;
};
```

要素	説明
data.vx	移動速度 X[m/s]
data.vy	移動速度 Y[m/s]
data.va	回転速度[rad/s]

３．４．３． 設定ファイル

本モジュールでは、特殊なドライバ、ライブラリを使用しておらず、使用するにあたって、OpenRTM-aist-1.0.0-RELEASE・RTSystemEditor の標準的な設定で問題はなく、新たにドライバ、ライブラリなどのインストールは必要ない。

３．５．　ObstacleAvidance（回避行動コンポーネント）

３．５．１．　機能概要

本 RTC は目標車体速度にて障害物との衝突判定を行ない、回避速度を出力するコンポーネントである。回避動作コンポーネントは、入力された車体速度に対して、障害物と衝突する可能性があるか判定し、衝突する可能性がある場合、回避可能な車体速度を計算し、出力する。回避経路の計算には、下図のように車体のキネマティクスを考慮し、可能な速度、角速度にて、決められた加速度、角加速度にて停止可能であるかを判定し、安全かつ目標速度に近い車体速度を出力とする。これにより、無理のない、スムーズな回避が可能である。

本モジュールを利用するためには、目標車体速度を与えるモジュール、センサ情報から障害物情報を生成するモジュール、車体速度に応じて移動ロボットを制御するモジュールが別途必要である。

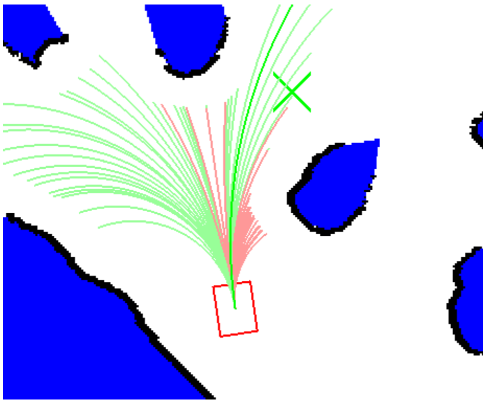

図　3-6　キネマティクスを考慮した回避経路の計画例

本 RTC に車体速度情報と、障害物情報が入力されると、目標車体速度にて走行した場合の軌跡で障害物との衝突チェックを行ない、衝突する場合ぶつからない速度に方向を調整し出力する。衝突しない場合、そのままの速度が出力され、衝突が予想される場合、衝突しない安全な回避車体速度が出力される。この際、回避経路はコンフィギュレーションパラメータ speed _weight および turn_weight の大小に応じて、減速を優先するか、回避を優先するかが選ばれる。回避を優先する場合、speed_weight を大きくすることにより、速度を維持しつつ回避する経路が選ばれ、減速を優先する場合、turn_weight を大きくすることにより、角度を維持しつつ、速度を減速していく動作が生成される。

３． ５． ２． 動作環境

本 RTC の動作環境（動作 OS、RT ミドルウェア、開発環境など）について記述する。

動作 OS	Linux（Ubuntu10.04 LTS）
開発言語	C/C++
RT ミドルウェア／バージョン	OpenRTM-aist-1.0.0-RELEASE
依存パッケージ	なし

３． ５． ３． ポート情報

本 RTC のポート情報を記述する。

A) データポート（InPort）

名称	型	データ長	説明
in_vel	IIS:: TimedVelocity2D	1	目標車体速度
obstacles	TimedObstacles	障害物数	障害物情報

B) データポート（OutPort）

名称	型	データ長	説明
out_vel	IIS::TimedVelocity2D	1	回避車体速度

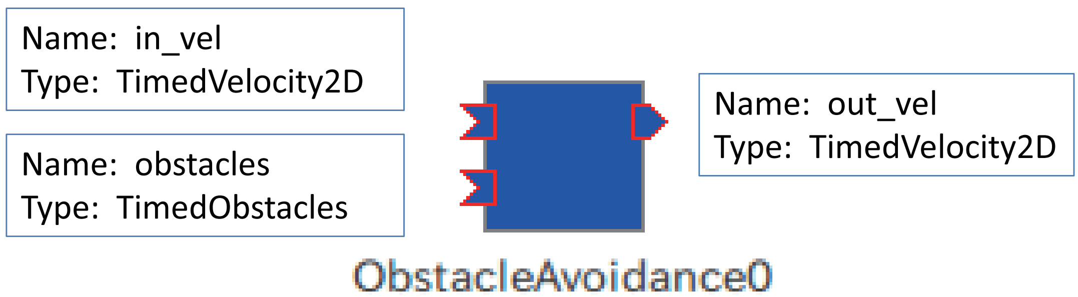

図 3-7 ObstacleAvoidance 単体モジュール構成図

３．５．４． コンフィグレーション

コンフィグレーション設定について記載する。

名称	型	デフォルト値	説明
body_radius	double	0.1	ロボットの半径[m]
speed_weight	double	0.3	回避時の速度に関する重み係数
turn_weight	double	0.3	回避時の角速度に関する重み係数
length_weight	double	0.3	走行距離に関する重み計数
interval_length	double	0.3	衝突判定処理を行う間隔[m]
space	double	0.5	車間距離
acceleration	double	0.1	減速度[m/s²]

３．５．５． 入出力データフォーマット

以下に動作およびデータ型詳細について示す。移動サブワーキンググループの共通インタフェース仕様については、「移動 SWG 共通 IF 案 101008.pdf」を参照のこと。

1) 動作

入力された車体速度から、数秒間の軌跡を推定し障害物と衝突する場合、設定した加速度、角加速度の範囲内で可能な速度の組み合わせに対し走行経路を推定し、その中で最適な経路を選択し、出力する。経路の選択においては、パラメータの変更により、速度優先、方向優先、安全優先を調整することが可能である。

2) データ型詳細

A) 入力：in_vel

型：IIS::TimedVelocity2D

```
struct TimedVelocity2D {
    RTC::Time tm;
    sequence<long> id;
    RTC::Velocity2D data;
    sequence<double> error;
};
```

要素	説明
data.vx	移動速度[m/s]
data.va	回転速度[rad/s]

B）入力：obstacles
型：TimedObstacles

```
struct TimedObstacles{
    Time tm;
    sequence<TimedObstacle> obstacles;
};
```

要素	説明
obstacles[id].x	障害物 X 位置
obstacles[id].y	障害物 Y 位置

※id は障害物数

C）出力：out_vel
型：IIS::TimedVelocity2D

```
struct TimedVelocity2D {
    RTC::Time tm;
    sequence<long> id;
    RTC::Velocity2D data;
    sequence<double> error;
  };
```

要素	説明
data.vx	移動速度 X[m/s]
data.vy	移動速度 Y[m/s]
data.va	回転速度[rad/s]

３．５．６． 設定ファイル

本モジュールでは、特殊なドライバ、ライブラリを使用しておらず、使用するにあたって、OpenRTM-aist-1.0.0-RELEASE・RTSystemEditor の標準的な設定で問題はなく、新たにドライバ、ライブラリなどのインストールは必要ない。

４．　特記事項

本モジュールをご利用される場合には、 以下の記載事項・条件にご同意いただいたものとします。

- 本モジュールは独立行政法人 新エネルギー･産業技術総合開発機構（NEDO 技術開発機構）の「次世代ロボット知能化技術開発プロジェクト」（平成 20 年～平成 23 年度）において、評価を目的として構成されたものである。
- LRF キャプチャコンポーネントの著作権は奈良先端科学技術大学院大学 情報科学研究科 ロボティクス講座が、障害物検知コンポーネント、衝突判定コンポーネント、回避行動コンポーネントの著作権は東北大学 大学院情報科学研究科 田所研究室が、障害物モニタコンポーネントの著作権は産業技術総合研究所が所有しています。
- ドキュメントに情報を掲載する際には万全を期していますが、それらの情報の正確性またはお客様にとっての有用性等については一切保証いたしません。
- 利用者が本モジュールを利用することにより生じたいかなる損害についても一切責任を負いません。
- 本モジュールの変更、削除等は、原則として利用者への予告なしに行います。また、止むを得ない事由により公開を中断あるいは中止させていただくことがあります。
- 本モジュールの情報の変更、削除、公開の中断、中止により、利用者に生じたいかなる損害についても一切責任を負いません。

【連絡先】
RTC 再利用技術研究センター
〒101-0021　東京都千代田区外神田 1-18-13　秋葉原ダイビル 1303 号室
Tel/Fax：03-3256-6353　E-Mail：contact@rtc-center.jp