RTミドルウェアコンテスト

「追従中の人の軌跡を予測するRTC」ユーザーマニュアル

芝浦工業大学 知能機械システム研究室 加藤宏一朗

目次

1. 追従中の人の軌跡を予測するRTC概要

1.1 はじめに

近年，サービスロボットの市場が急激に拡大すると予測されている．人と共存するサービスロボットは，高齢者の介護や商業施設での道案内，外国人との対話など，直接人と関わる場合が多く，これらの場合では，できるだけ人の近くにロボットがいることが好ましい．そのためにロボットには人物追従の機能は必要不可欠である．人とロボットがやりとりを行う際にロボットは人の動きを見て動く必要があり，ロボットが決まった経路を移動するよりも，ロボットが自由に動く方が適しているからである．本研究室では，RTミドルウェアを用いて追従ロボットの研究が行われてきた．追従ロボットに必要不可欠な機能として，特定の人物（ターゲット）を追従すること，ターゲットを見失った際に正確に人物を再検出するロバスト性を持つこと，そのターゲットを再追従することが挙げられる．そこで，追従精度の向上を目的に予め機械学習で学習した予測器を用いて，追従中にリアルタイムでターゲットの軌跡を予測するRTCを開発した．

1.2 開発・動作環境

- PC ①
    - OS：Windows7
    - OpenRTM-aist1.2.0(32bit版)
    - CMake
    - OpenNI
- PC ②
    - OS：Windows 10
    - OpenRTM-aist1.2.0(64bit版)
    - Python 3.7.2 (64bit版)

1.3 使用機器

1.3.1 移動台車

本システムでは図1に示す移動ロボットを用いる．移動台車はVECTOR株式会社のコンシェルジュ，頭部にDepthセンサ，腹部には測域センサが搭載されている．

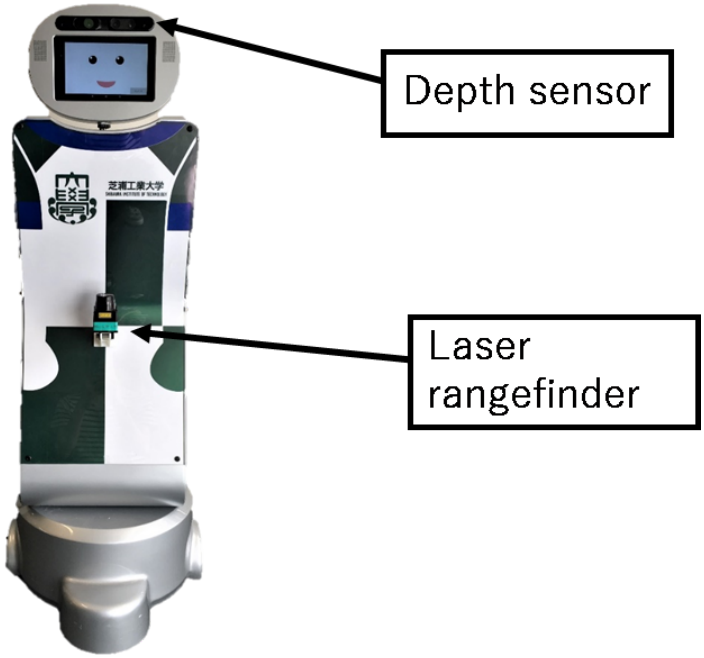

図1-1 移動台車

1.3.2 測域センサ

測域センサは北陽電機株式会社のURG-04LX-UG01^1を用いた．図1-2に外観，表1-1に主な仕様を示す．

図1-2 測域センサ

表 1-1 URG-04LX-UG01の主な仕様

項目	仕様
光源	半導体レーザ：λ=785 [nm]（FDAレーザ安全クラス1）
測距範囲	距離：0.02-5.6[m] 角度：240[°]
測距精度	0.06-1[m]：±30[mm], 1-4[m]：距離の3[%]
測距分解能	約1[mm]
角度分解能	ステップ角：約0.36[°]

項目	仕様
走査時間	100[ms/scan]

1.3.3 Depthセンサ

DepthセンサはASUS社の Xtion Pro LIVE®^2を用いた．OpenNIを用いて人の骨格情報を取得する．図1-3に外観，表1-2に主な仕様を示す．

図1-3 Depthセンサ

表 1-2 Xtion PRO LIVE の主な使用

項目	仕様
センサー	RGBセンサ，深度センサ，ステレオマイク
深度センサ解像度	640×480ドット (VGA) /30fps 320×240ドット (QVGA) /60fps
センサ有効範囲	水平58[°]，垂直45[°]，対角70[°]
センサ有効距離	0.8m-3.5m

2. 本システムの各RTCの概要と仕様

本システムは，"Kinect RTC"，"URG RTC"，"Concierge_Type3_verOLD","object_tracking_concierge RTC"，"TrajectoryPrediction RTC"で構成されている．図2-1に本システムのRTC図，表2-1に概要を示す．

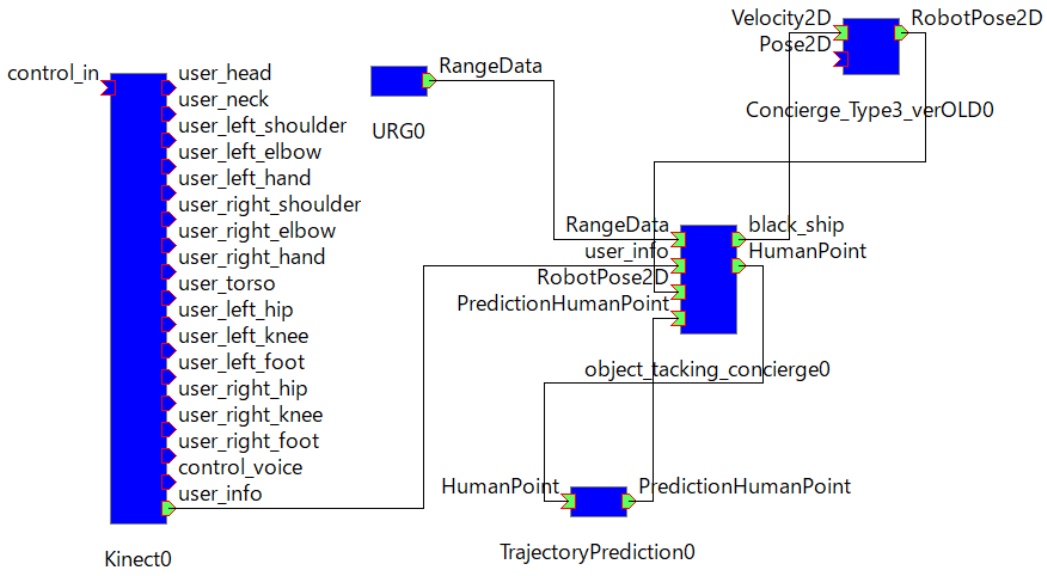

図2-1 RTC図

表2-1 各RTC概要

RTC名	説明
Kinect	xtionから人の座標を取得するRTC
URG	測域センサから値を取得するRTC
Concierge_Type3_verOLD	移動台車を動かすRTC
object_tracking_concierge	センサからの値を受け取り指令を出すRTC
Prediction	今回開発した人の軌跡を予測するRTC

2.1 各RTCの仕様

- Kinect RTC
  本RTCは先述したXtion Pro Liveのセンサデータから OpenNIを使用して人の座標を取得し，出力するRTCである．図2-2にRTCを示す．人の部位名のアウトポートから人の座標(x, y, z)を出力する．また，人のid，右手，右ひじ，右肩の情報を文字列にしたデータを user_info から出力している．今回はこのOutportを用いる．

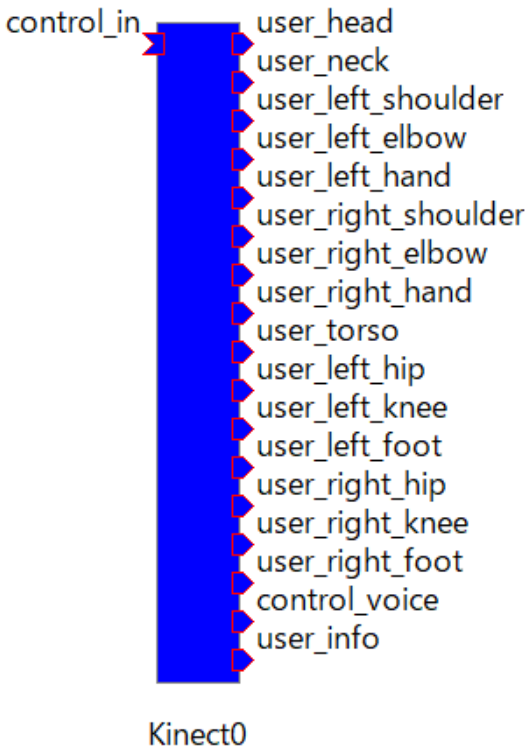

図2-2 Kinect RTC

- URG RTC
  本RTCは，先述したURG-04LXのセンサデータを取得し，出力するRTCである．ロボット周囲の障害物と人の検知に使用する．Xtionの方で何かのエラーで人を見失った際にはURGからの人のデータで追従を行う．

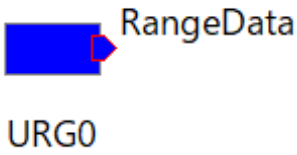

図2-3 URG RTC

- Concierge_Type3_verOLD RTC
  本RTCは，object_tracking_concoergeから速度指令を受け取り，移動台車を動かすRTCである．また，Outportからオドメトリを出力する．

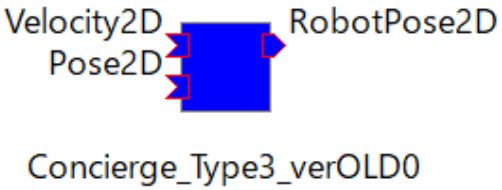

図2-4 Concierge_Type3_verOLD RTC

- object_tracking_concierge RTC
  本RTCは，kinect RTC から人の位置情報，URG RTC からRangeデータを受け取り，それらのデータ

を統合し，人との距離が一定になるように移動台車に速度指令を送る．また，移動台車からオドメトリを受け取り，ワールド座標系の人の座標を計算し，軌跡予測RTCに座標を出力する．

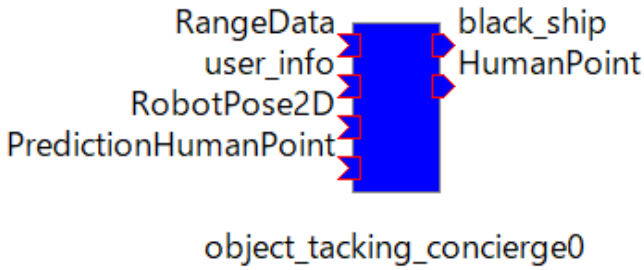

図2-5 object_tracking_concierge RTC

- TrajectoryPrediction RTC
  本RTCは今回開発した人の移動軌跡を予測するRTCである． object_tracking_concierge RTCからワールド座標系の人の位置座標を受け取り，それをもとに軌跡の予測をする．詳細は次章にて解説する．

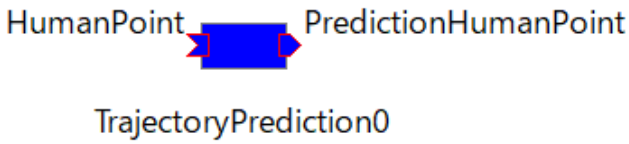

図2-6 TrajectoryPrediction RTC

3. 軌跡予測RTC(TrajectoryPrediction RTC)

本RTCは今回新規に開発したRTCである．東京女子大学の加藤研究室^3がSocial-LSTM^4を用いて予測器を構築した．機械学習した予測器で追従対象者の軌跡を予測して出力する．InportのHumanPointは人の座標を受け取るポートであり，object_tracking_conciergeから追従中の追従対象者の位置座標を受け取る．OutportのPredictionHumanPointは予測した人の座標を出力する．今回は学習に用いたデータセットが〜のため，ワールド座標系の人の座標をもとに予測をする．

現在の仕様は.txtファイル経由でデータの受け取り，出力を行っている．まずRTCが人の位置座標を受け取り，`directory/nantoka.txt`に人のデータを書き込む．そのファイルから最新の10フレーム分を用いて予測器にて予測を行い `dire/kekka.txt`に予測した10フレーム分のデータを出力する．その`kekka`ファイルからデータを出力する．

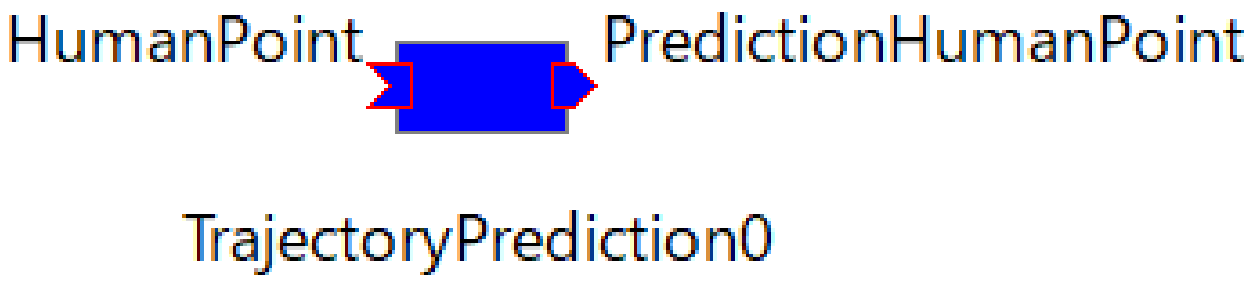

図3-1 TrajectoryPrediction RTC

表 3-1 TrajectoryPrediction RTCポート

In/Out	Port名	データ型	機能	データの例
In	HumanPoint	TimedDoubleSeq	人の座標を受け取る	HumanPoint.data.[0] = 人のx座標 HumanPoint.data.[1] = 人のy座標
Out	PredictionHumanPoint	TimedDoubleSeq	予測した人の座標を出力する	PredictionHumanPoint.data.[i] = 人のx座標(i=1 - 10の奇数) PredictionHumanPoint.data.[i] = 人のy座標(i=1 - 10の偶数)

3.2 予測結果

今回の予測器を用いて簡単な実験を行ったところ以下のようになった．10フレーム分の人の座標データからその先10フレーム分を予測する．実際に予測をロボットに適応する際は5フレーム分を受け取り，その座標をロボットに追従させる．安全面を考慮して，予測結果が人を見失った座標から大きく離れていた場合はその予測を適応しない．

4. 本システムの使用方法

4.1 ハードウェアの準備

本システムではPCを2台使用する．移動ロボット，センサーに接続するPCと予測器を動かすPCとなっている．前者をPC①，後者をPC②とする．
PC①にURG，Xtion，移動台車を接続する．

4.2 動作環境

以下に本稿で使用するシステムの動作環境を示す．

- PC①
  - OS：Windows 7
  - **OpenNIが使用できるPC**
- PC②
  - OS：Windows 10
  - **OpenRTM 1.2.0 64bit版，Python3系 64bit版が使用できるPC**

また，PC②のTrjectoryPrediction RTCを動作させるにはpip等で以下のモジュールをインストールする必要がある．また，pytorchはwindows上のPython2系およびPython 32bit版には対応していない．

- numpy
- pandas
- pytorch

4.3 RTCのダウンロード

GithubからPC①にobject_tracking_concierge，PC②にTrajectoryPredictionをダウンロードする．
object_tracking_concierge RTCはC++言語で書かれているため，ビルドを行う必要がある．

4.4 システムの起動

- PC①，PC②共通 Start Naming Service とeclipseを起動する．ワークスペースの選択ではRTCのフォルダがあるワークスペースを選択する．また，2台のPCが同ネットワークに接続されている必要がある．

- PC① PC①で動作させるRTCはC++で書かれているため，exeファイルから起動させる．

RTC名	起動ファイル
Kinect	KinectComp.exe
URG	URGComp.exe
Concierge_Type3_verOLD	Concierge_Type3_verOLDComp.exe
object_tracking_concierge	object_tracking_conciergeComp.exe

- PC② TrajectoryPrediction RTCはPython言語で書かれているため，コマンドプロンプトからPythonファイルのあるディレクトリに移動して以下のように実行するか，図なんとかのようにファイルを選択して実行する．

```
Python TrajectoryPrediction.py
```

4.5 RTCの接続

4.5.1 PC間の接続

PC①またはPC②のEclipseでネームサーバーの追加からもう一方のPCのIPアドレスを入力して接続する．

4.5.2 各RTCの接続

4.5.1でネームサーバーを追加したPCで，4.4で起動したRTCをSystem Diagram上にドラッグ&ドロップし，各RTCのポートを図2-1のように接続する．

4.6 システムの起動

各コンポーネントをアクティベートする．PC①上でPrime Sense USer Tracker ViewerとURG_Dataのウィンドウが起動するのでの表示を図4-1のように表示させる．URG_Dataのウィンドウはobject_tracking_concierge RTC内でウィンドウプロシージャを使用して表示させているのでこのウィンドウが前面にない場合，動作しない．

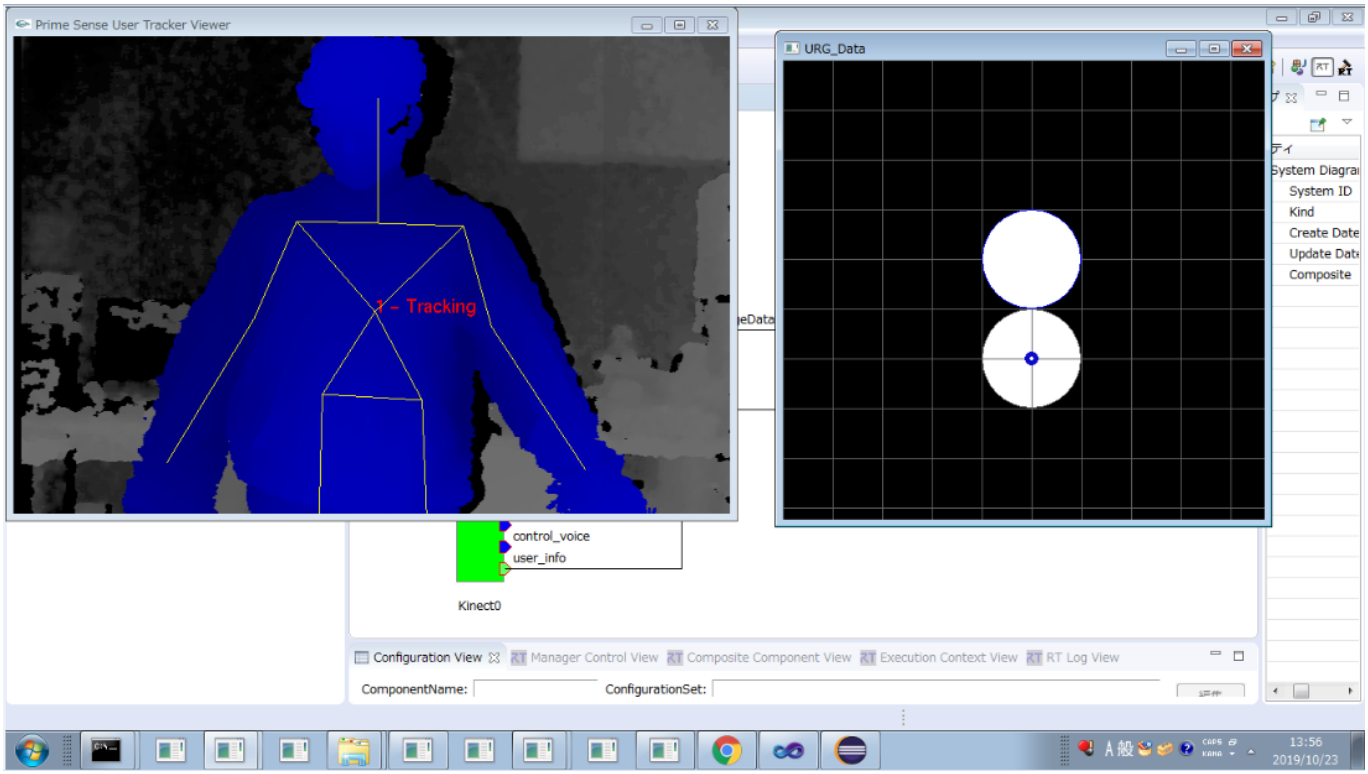

図4-1 表示画面

4.7 追従の開始

右手より右ひじが上かつ，右ひじより右肩が上のポーズ(図4-2)を認識したタイミングで追従を開始する．追従を終わらせたい場合は再度右手を上げる．今回開発した TrajectoryPrediction RTCは曲がり角，障害物の回避後などで人を見失った場合(Xtion，URGからの人の座標が更新されなくなった場合)に動作する．

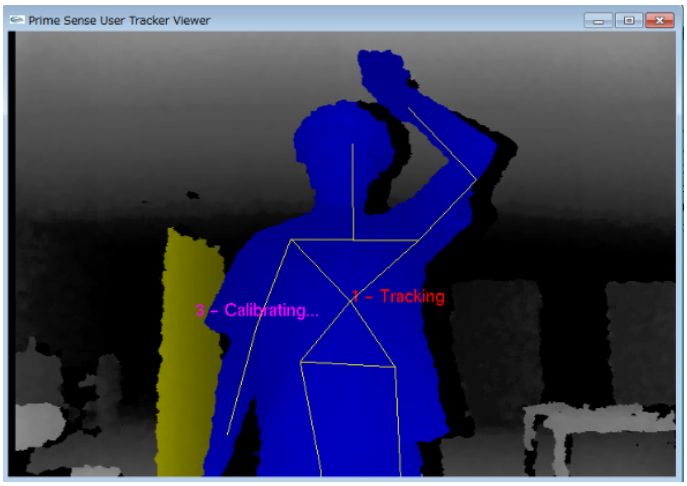

図4-2 追従開始/終了のポーズ

5. 参考文献