RTコンポーネント操作マニュアル

分散レーザレンジファインダの
キャリブレーション支援
コンポーネント群

平成19年12月1日版

東京大学生産技術研究所

橋本研究室

佐々木 毅

目 次

1　本コンポーネント群の概要

1.1　開発の背景

近年、センサやアクチュエータを空間に分散配置することで実現される知能化空間に関する研究が広がりを見せています。知能化空間では、空間に埋め込まれたセンサが互いに情報をやり取りすることによって空間内の事象を認識し、その情報に基づいてアクチュエータがユーザに対して情報的・物理的サービスを提供します。これまで、人間の行動の観測、移動ロボットの行動支援など様々な知能化空間のアプリケーションが提案されています。

しかし、こうしたシステムを構築していく際の問題の 1 つとして、センサのキャリブレーションが挙げられます。各センサの座標系（ローカル座標系）において得られたデータを知能化空間の座標系（ワールド座標系）へ変換するためにはキャリブレーションが必要となりますが、多数のセンサをキャリブレーションするには多くの手間を要します。そこで、この作業の効率化を目的とし、分散センサのキャリブレーション支援コンポーネント群を開発することといたしました。

知能化空間に配置されるセンサとしては、カメラやマイクロフォン、圧力センサ等、様々なセンサが考えられますが、中でもレーザレンジファインダには、(1) 設置が比較的容易である、(2) 知能化空間の基本機能の 1 つである物体トラッキングにおいても対象に特別なタグを持たせる必要がない、(3) 空間の地図も獲得することができる、といった利点があり、主要なセンサの１つとして期待されています。そのため、本コンポーネント群では特にレーザレンジファインダの位置・姿勢のキャリブレーションに着目しております。

1.2　開発したコンポーネント群

キャリブレーション機能の実証のため、キャリブレーションコンポーネント (LRFCalibration) を始め、以下のようなコンポーネント群を開発しました。それぞれのコンポーネントの詳細については 2 章を参照してください。

- レーザレンジファインダコンポーネント (LRFComponent)

  北陽電機株式会社製のレーザレンジファインダ URG-04LX を RT コンポーネント化したもの。

- 移動体トラッキングコンポーネント (SimpleTracker)

  レーザレンジファインダのスキャンデータから移動物体の位置を出力する。

- キャリブレーションコンポーネント (LRFCalibration)

  同一物体の 2 つの座標系での位置を入力として受け取り、その対応点の情報から 2 つの座標系の位置・姿勢の関係を出力する。

- 座標変換コンポーネント (CoordTrans2D)

  キャリブレーションパラメータに基づいてセンサ座標系から基準座標系への座標変換を行う。

- 入力コンポーネント (ConsoleIn2)

  コンソールから入力した値を順に 2 つの OutPort に出力する。

1.3　キャリブレーションコンポーネント (LRFCalibration) の使用例

キャリブレーションコンポーネント (LRFCalibration) は以下のような使い方ができます。

- キャリブレーションオブジェクトを用いた絶対的な位置・姿勢の手動キャリブレーション（※）
- 移動物体を用いたレーザレンジファインダ間の相対的な位置・姿勢の自動キャリブレーション（※）
- 移動ロボットを用いた絶対的な位置・姿勢の自動キャリブレーション

- 天井カメラの位置 · 姿勢のキャリブレーション

(※) は新たなコンポーネントを開発しなくとも本コンポーネント群のみで実現可能な機能です。機能及び RTCLink を用いたシステム構築の詳細については 3 章を参照してください。

1.4 開発環境

開発環境は以下の通りです。

- OS: Ubuntu Linux 7.0.4
- RT ミドルウェア: OpenRTM-aist-0.4.1-RELEASE
- コンパイラ: gcc 4.1.2
- CORBA: omniORB 4.0.7
- ACE (The ADAPTIVE Communication Environment): ACE 5.4.7-12
- Eclipse: Eclipse 3.2
- Java 実行環境: Sun Java 1.5.0-11-1

2 各コンポーネントの説明

2.1 レーザレンジファインダコンポーネント (LRFComponent)

LRFComponent は北陽電機株式会社製のレーザレンジファインダ URG-04LX を RT コンポーネント化したものです。アクティブ化すると距離データと計測範囲の情報を OutPort に出力します。距離データの単位は URG04-LX の出力形式と同じ mm です。

また、測定パラメータを設定するための Configuration 変数と、最大計測距離などのセンサ情報を他のコンポーネントに提供するためのサービスポートが用意されています。

- InPort

  なし

- OutPort

名称	型	説明
ScanData	TimedShortSeq	センサが獲得した距離データ。単位は mm。
StartPoint	TimedShort	スキャンの開始点（ステップ）。計測範囲の情報の 1 つ。詳しくは URG の仕様書を参照。
EndPoint	TimedShort	スキャンの終了点（ステップ）。計測範囲の情報の 1 つ。詳しくは URG の仕様書を参照。

- Configuration 変数

名称	型	デフォルト値	説明
DeviceName	char*	“/dev/ttyACM0”	デバイスが接続されたポートの名前。詳しくは URG の仕様書を参照。アクティブ状態での変更は無効（反映されない）。
ScanRate	int	19200	通信速度。単位は bps。詳しくは URG の仕様書を参照。アクティブ状態での変更は無効（反映されない）。
ScanStart	int	0	スキャンの開始点（ステップ）。有効範囲は [0,768]。詳しくは URG の仕様書を参照。
ScanEnd	int	768	スキャンの終了点（ステップ）。有効範囲は [0,768] で、ScanStart よりも大きい値。詳しくは URG の仕様書を参照。
ScanStep	int	1	ScanStart と ScanEnd の間のスキャンのステップ数。有効範囲は [1,99]。詳しくは URG の仕様書を参照。

- サービスポート

  InfoPort (provider)

関数名	引数	戻り値の型	説明
getMaxRange	なし	short	センサの最大計測可能距離を返す。単位は mm。
getMinAngle	なし	short	最小ステップ（ステップ 0）に対応する角度を返す。単位は deg。
getMaxAngle	なし	short	最大ステップに対応する角度を返す。単位は deg。
getMaxStep	なし	short	最大ステップを返す。

2.2 移動体トラッキングコンポーネント (SimpleTracker)

SimpleTracker は、レーザレンジファインダのスキャンデータから移動物体の位置を出力するものです。アクティブ状態のときに LRFComponent から観測データを受け取ると、まず背景情報（静止物体領域の情報）を獲得します。そのため、アクティブ化を行うときは観測領域に人間などの移動物体が入らないように注意してください。背景情報の獲得後は、受信した距離データから背景と異なる部分を見つけ出し、その物体のスキャン平面上での位置を出力します。観測領域内に複数の移動物体がある場合には最大の大きさを持つ物体の位置を出力します。出力の単位は m です。(URG04-LX の出力形式が mm ですので、移動体位置の出力に際して 1/1000 倍しています。したがって、cm や m で出力するセンサを接続した場合は m にはなりません。）非アクティブ化すると背景情報等は失われ、アクティブ化すると再び背景情報の獲得から処理が行われます。

なお、このコンポーネントでは、アクティブ状態で InPort の ScanData のデータ長、StartPoint 及び EndPoint の値が変化した場合、移動体トラッキング処理を中断します。LRFComponent の測定パラメータを変化させたい場合には、このコンポーネントを非アクティブ状態にしてから行ってください。

- InPort

名称	型	説明
ScanData	TimedShortSeq	LRFComponent が出力した距離データ。単位は mm を想定。
StartPoint	TimedShort	スキャンの開始点（ステップ）。LRFComponent が出力した計測範囲の情報の 1 つ。
EndPoint	TimedShort	スキャンの終了点（ステップ）。LRFComponent が出力した計測範囲の情報の 1 つ。

- OutPort

名称	型	説明
X	TimedDouble	観測領域内で最大の大きさをもつ移動物体の位置の x 座標。単位は m を想定（Scan Data の 1/1000）。
Y	TimedDouble	観測領域内で最大の大きさをもつ移動物体の位置の y 座標。単位は m を想定（Scan Data の 1/1000）。

- Configuration 変数

  なし

- サービスポート

  InfoPort (consumer)

  提供される関数については 2.1 節レーザレンジファインダコンポーネント (LRFComponent) のサービスポートを参照。

2.3 キャリブレーションコンポーネント (LRFCalibration)

LRFCalibration は、同一物体の 2 つの座標系での位置を入力として受け取り、その対応点の情報から次式で与えられる座標変換における 2 つの座標系の位置 · 姿勢の関係 (T_x, T_y, θ) を出力します。

$$\begin{aligned} x_g &= \cos\theta x_l - \sin\theta y_l + T_x \\ y_g &= \sin\theta x_l + \cos\theta y_l + T_y \end{aligned} \tag{1}$$

ここで、$(x_g,\ y_g), (x_l,\ y_l)$ はそれぞれ基準となる座標系とセンサ座標系における位置を表しています。入力の単位は任意ですが、全ての入力で統一する必要があります。出力の単位は、並進量は入力と同じ、回転角は rad です。

まず Configuration から、キャリブレーションに用いる対応点の数 data_num を設定してください。これは最低でも 2 組は必要です。設定はアクティブ化前でも後でも可能ですが、アクティブ化後に設定を変更した場合にはそれまでの対応点のデータは初期化され失われます。アクティブ化状態で入力を待ちます。対応点が設定した data_num 個だけ集まると計算が行われます。

- InPort

名称	型	説明
GlobalX	TimedDouble	物体の基準となる座標系での位置の x 座標。単位は任意だが他の入力と統一する必要がある。
GlobalY	TimedDouble	物体の基準となる座標系での位置の y 座標。単位は任意だが他の入力と統一する必要がある。
LocalX	TimedDouble	物体のセンサ座標系での位置の x 座標。単位は任意だが他の入力と統一する必要がある。
LocalY	TimedDouble	物体のセンサ座標系での位置の y 座標。単位は任意だが他の入力と統一する必要がある。

- OutPort

名称	型	説明
Tx	TimedDouble	基準座標系に対するセンサ座標系の x 方向の並進量。単位は入力と同じ。
Ty	TimedDouble	基準座標系に対するセンサ座標系の y 方向の並進量。単位は入力と同じ。
Theta	TimedDouble	基準座標系に対するセンサ座標系の回転角度。単位は rad。

- Configuration 変数

名称	型	デフォルト値	説明
data_num	int	0	キャリブレーションに用いる対応点の数。最低でも 2 組は必要。アクティブ状態で変更した場合、それまでの対応点のデータは初期化される。

- サービスポート

  なし

2.4 座標変換コンポーネント (CoordTrans2D)

CoordTrans2D は、キャリブレーションパラメータと式 (1) を用いて与えられた位置をローカル座標系から基準座標系へ変換します。入力座標と並進量の単位は任意ですが、全て統一する必要があります。回転角度の単位は deg です。出力の単位は入力と同じです。

- InPort

名称	型	説明
Xin	TimedDouble	センサ座標系の位置の x 座標。単位は任意だが他の入力と統一する必要がある。
Yin	TimedDouble	センサ座標系の位置の y 座標。単位は任意だが他の入力と統一する必要がある。
Tx	TimedDouble	基準座標系に対するセンサ座標系の x 方向の並進量。単位は任意だが他の入力と統一する必要がある。
Ty	TimedDouble	基準座標系に対するセンサ座標系の y 方向の並進量。単位は任意だが他の入力と統一する必要がある。
Theta	TimedDouble	基準座標系に対するセンサ座標系の回転角度。単位は rad。

- OutPort

名称	型	説明
Xout	TimedDouble	基準座標系に変換した位置の x 座標。単位は入力と同じ。
Yout	TimedDouble	基準座標系に変換した位置の y 座標。単位は入力と同じ。

- Configuration 変数

  なし

- サービスポート

  なし

2.5 入力コンポーネント (ConsoleIn2)

ConsoleIn2 は、OpenRTM-aist のホームページ (http://www.is.aist.go.jp/rt/OpenRTM-aist/) にサンプルとして取り上げられている ConsoleIn コンポーネントを 2 出力にし、OutPort の変数の型を TimedDouble にしたものです。アクティブ化すると数値の入力を促すメッセージがコンソールに表示されます。コンソールから数値を入力すると、その数値を順に 2 つの OutPort に出力します。

- InPort

  なし

- OutPort

名称	型	説明
out	TimedDouble	コンソールから最初に入力された数値。
out2	TimedDouble	コンソールから 2 番目に入力された数値。

- Configuration 変数

  なし

- サービスポート

  なし

3 本コンポーネント群の使用方法

3.1 RtcLink によるシステム構築の手順

RTCLink を用いたシステム構築の手順は以下の通りです。

[1] ネームサーバの起動

[2] コンポーネントの起動

[3] RTCLink の起動

[4] ネームサーバへの接続

[5] システムの構築と実行

これらの手順はどのようなシステムでも共通ですので、操作方法は OpenRTM-aist のホームページ (http://www.is.aist.go.jp/rt/OpenRTM-aist/) を参照してください。以下の節では手順 [5] に関して、本コンポーネント群の具体的な使用例を説明します。

3.2 キャリブレーションオブジェクトを用いた絶対的な位置 · 姿勢の手動キャリブレーション

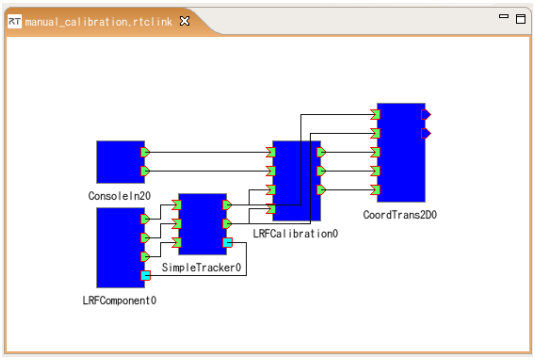
レーザレンジファインダで検知しやすい物体を環境内の既知の位置に順番に置き、その位置情報に基づいてレーザレンジファインダの絶対的な位置 · 姿勢を計算するというものです。精度が必要な場合はこの方法を使用してください。使用するコンポーネントの種類とその数は

- レーザレンジファインダコンポーネント (LRFComponent) · · · · · · 1 つ
- 移動体トラッキングコンポーネント (SimpleTracker) · · · · · · 1 つ
- キャリブレーションコンポーネント (LRFCalibration) · · · · · · 1 つ
- 入力コンポーネント (ConsoleIn2) · · · · · · 1 つ
- 座標変換コンポーネント (CoordTrans2D) · · · · · · 1 つ

の計 5 つです。座標変換コンポーネントは結果の確認用ですのでキャリブレーションを行うだけならなくてもかまいません。以下に使用手順を示します。

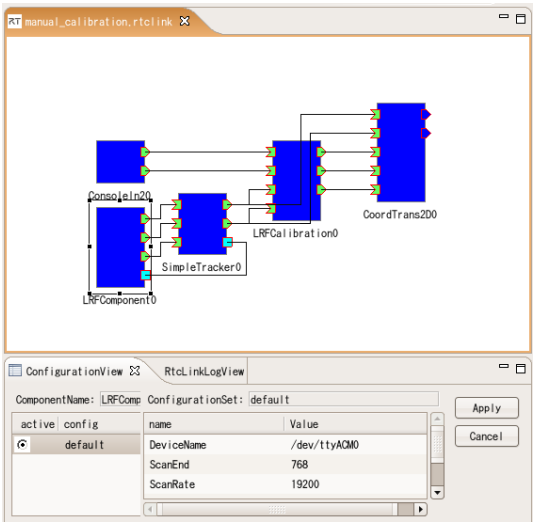
(1) システムの構築

システムエディタを用いて下図のようにコンポーネントを接続し、システムを構築してください。

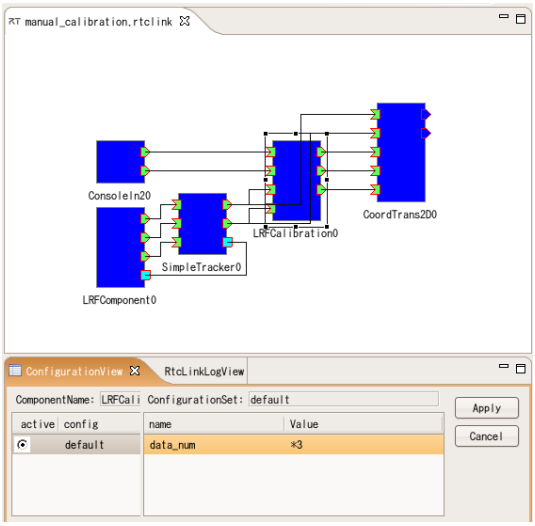


(2) Configuration 変数の設定

LRFComponent を選択し、ConfigurationView から測定パラメータ等を設定してください。通常はデフォルトのままでよいと思います。



LRFCalibration を選択し、ConfigurationView からキャリブレーションに用いる対応点の数 data_num を設定してください。最低でも 2 組は必要ですが、3 程度でよいと思います。

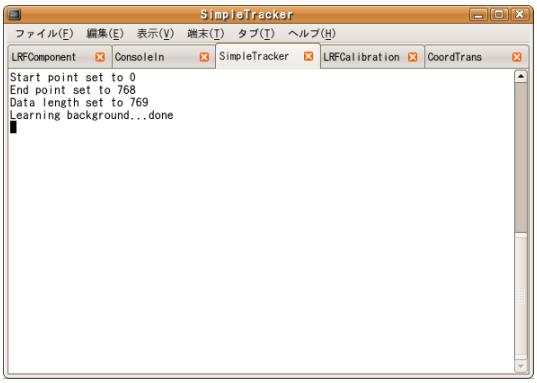


(3) コンポーネントのアクティブ化

ツールバーの All Activate のボタンをクリックするか、システムエディタ上の右クリックメニューから All Activate を選択し、全てのコンポーネントをアクティブ化してください。SimpleTracker は、最初に背景情報（静止物体領域の情報）を獲得するので、アクティブ化を行うときは観測領域に人間などの移動物体が入らないように注意してください。SimpleTracker のコンソールに

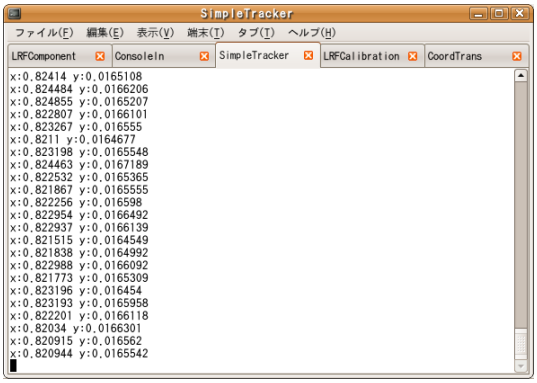
Learning background...done

と表示されれば背景情報の獲得は完了です。

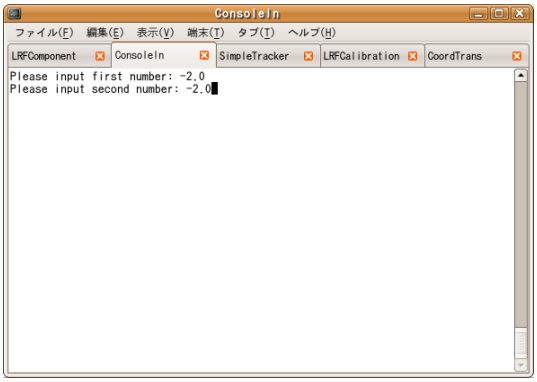


(4) キャリブレーション

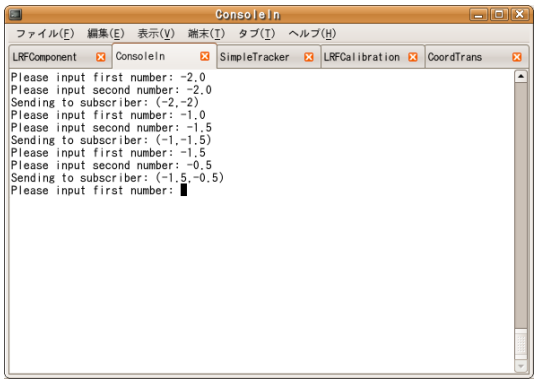
キャリブレーションオブジェクト（レーザレンジファインダで検知できればどんな物でもよいです）を観測領域内の既知の位置に置きます。SimpleTracker のコンソールに安定した出力が表示されていることを確認してください。

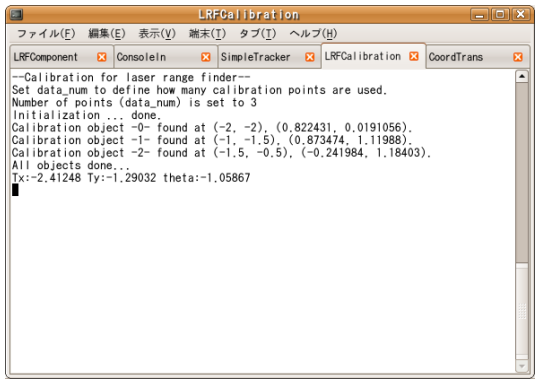


確認ができたら、ConsoleIn2 のコンソールからキャリブレーションオブジェクトの位置の座標を入力します。



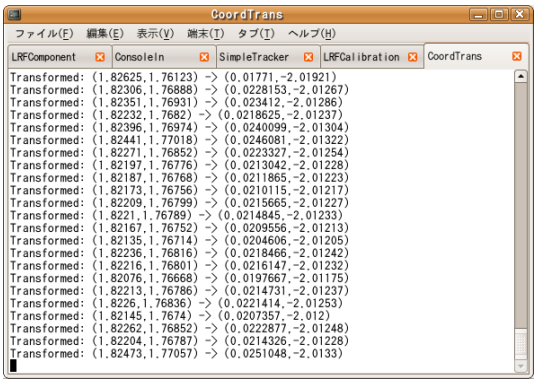
キャリブレーションオブジェクトの位置を変えて同じ作業を (2) で設定した data_num の回数だけこの作業を繰り返します。全ての入力が完了すると LRFCalibration のコンソールにキャリブレーション結果が表示されます。この結果は OutPort からも出力されますので、他のコンポーネントが利用することも可能です。





(5) キャリブレーション結果の確認

CoordTrans2D を利用している場合は結果の確認ができます。物体を既知の位置に置き、CoordTrans2D のコンソールに正しい結果が得られていることを確認してください。下図は $(0, -2.0)$ に物体を置いた場合の結果を示しています。



3.3 移動物体を用いたレーザレンジファインダ間の相対的な位置 · 姿勢の自動キャリブレーション

2 台のレーザレンジファインダの観測領域の重なりを利用し、一方のレーザレンジファインダを基準とした相対的な位置 · 姿勢を獲得するというものです。3.2 節の手動キャリブレーションと比べると精度は劣りますが、キャリブレーションオブジェクトを順に配置する必要がありませんので簡単にキャリブレーションを行うことができます。

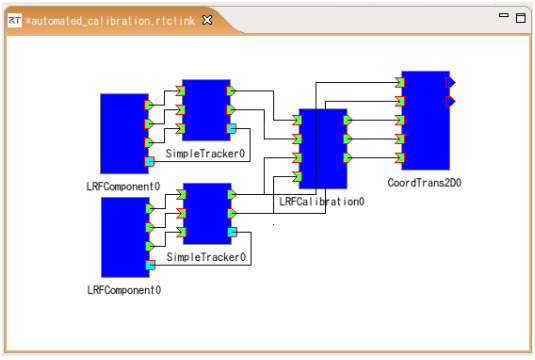
使用するコンポーネントの種類とその数は

- レーザレンジファインダコンポーネント (LRFComponent) · · · · · · 2 つ
- 移動体トラッキングコンポーネント (SimpleTracker) · · · · · · 2 つ
- キャリブレーションコンポーネント (LRFCalibration) · · · · · · 1 つ
- 座標変換コンポーネント (CoordTrans2D) · · · · · · 1 つ

の計 6 つです。座標変換コンポーネントは結果の確認用ですのでキャリブレーションを行うだけならなくてもかまいません。以下に使用手順を示します。

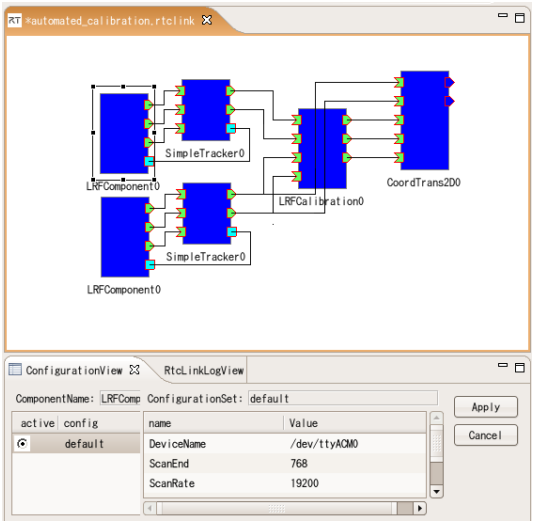
(1) システムの構築

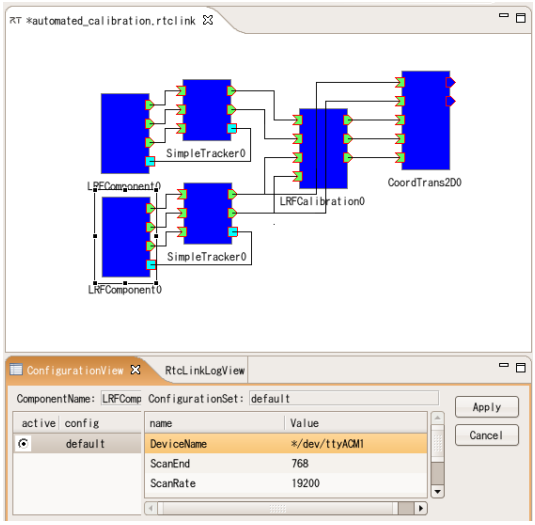
システムエディタを用いて下図のようにコンポーネントを接続し、システムを構築してください。



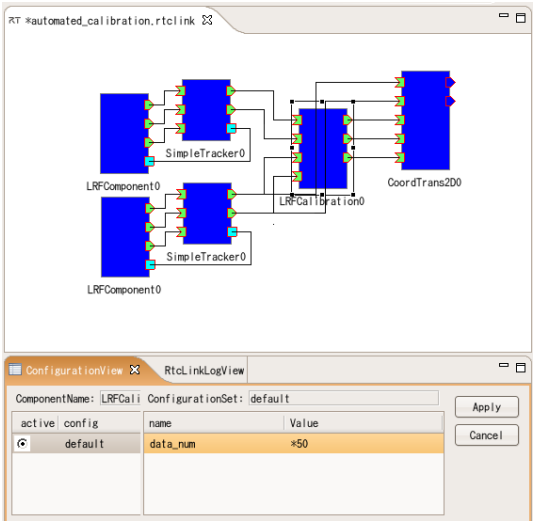
(2) Configuration 変数の設定

LRFComponent を選択し、ConfigurationView から測定パラメータ等を設定してください。（2 つあるのでそれぞれ設定してください。）通常はデフォルトのままでよいと思いますが、1 台のコンピュータに 2 台のレーザレンジファインダを接続している場合などは DeviceName を適切な値にしてください。





LRFCalibration を選択し、ConfigurationView からキャリブレーションに用いる対応点の数 data_num を設定してください。30 から 50 程度がよいのではないかと思います。

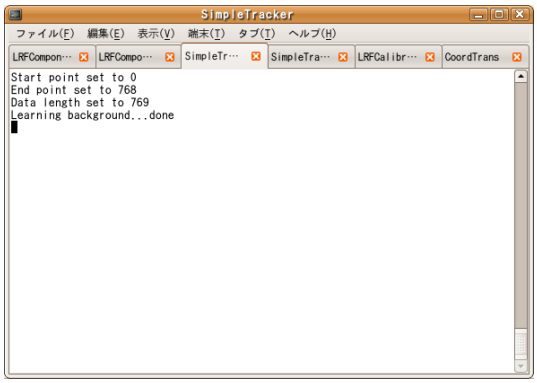


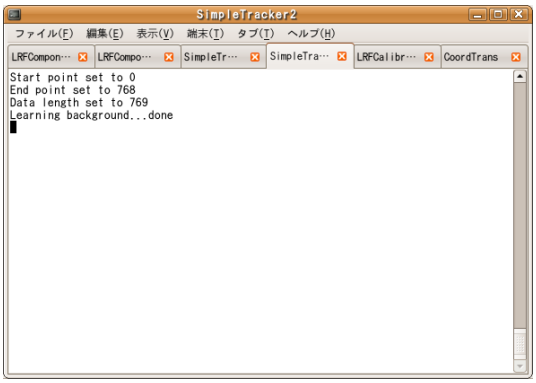
(3) コンポーネントのアクティブ化

ツールバーの All Activate のボタンをクリックするか、システムエディタ上の右クリックメニューから All Activate を選択し、全てのコンポーネントをアクティブ化してください。SimpleTracker は、最初に背景情報（静止物体領域の情報）を獲得するので、アクティブ化を行うときは観測領域に人間などの移動物体が入らないように注意してください。SimpleTracker のコンソールに

Learning background...done

と表示されれば背景情報の獲得は完了です。



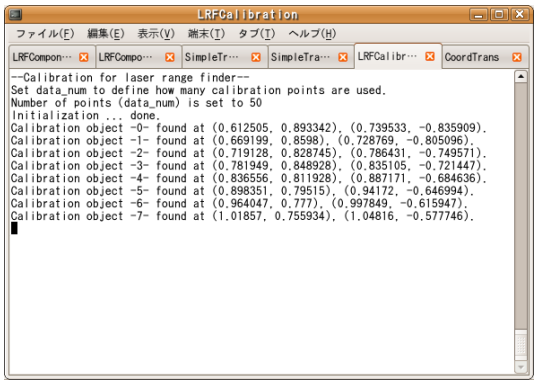


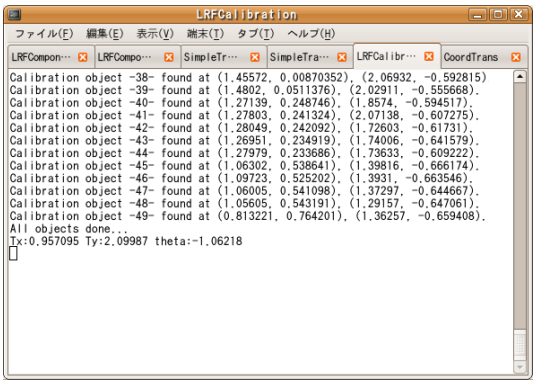
(4) キャリブレーション

2 台のレーザレンジファインダの観測領域の重なり部分で、人間やロボットなど物体を移動させてください。できるだけ広い範囲を移動させた方が結果はよくなります。

なお、レーザレンジファインダでは物体の輪郭しか獲得できませんので、物体の右側面と左側面といったように 2 台のレーザレンジファインダで異なる点を抽出することになります。移動物体が大きいと推定誤差も大きくなることがあるので注意してください。また、現在の SimpleTracker は観測領域内で最大の大きさを持つ物体の位置を出力します。複数の移動物体がいると対応付けを誤る可能性があるので、観測領域内の移動物体は 1 つのみにしておくことを推奨します。

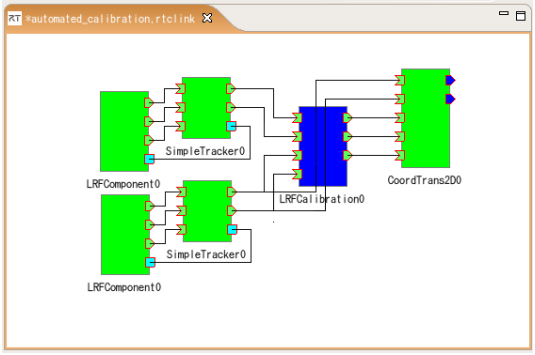
対応点のデータ (2) で設定した data_num 個だけ集まると LRFCalibration のコンソールにキャリブレーション結果が表示されます。この結果は OutPort からも出力されますので、他のコンポーネントが利用することも可能です。



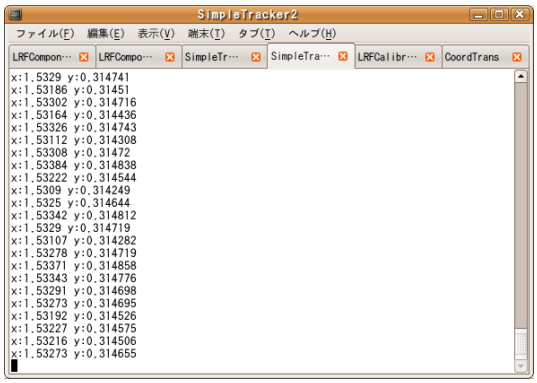


(5) キャリブレーション結果の確認

CoordTrans2D を利用している場合は結果の確認ができます。まず、LRFCalibration を非アクティブ化してキャリブレーション処理を停止させます。



次に、観測領域内の適当な位置に物体を置き、基準となる SimpleTracker（LRFCalibration の GlobalX, GlobalY に接続されているコンポーネント）のコンソールと CoordTrans2D のコンソールでほぼ同じ結果が得られていることを確認してください。



3.4 その他の使用例

今回実現した機能以外にも、他のコンポーネントと連携させることでキャリブレーションコンポーネント (LRFCalibration) を様々に利用することができます。ここでは、その他の使用例を示します。

- 移動ロボットを用いた絶対的な位置 · 姿勢の自動キャリブレーション

  3.3 節の自動キャリブレーションでは、移動物体の絶対的な位置情報がわからないため、キャリブレーションも相対的に行うことになります。しかし、例えば基準となるレーザレンジファインダの代わりに移動ロボットの自己位置推定モジュールから得られる位置情報を利用すれば、レーザレンジファインダの絶対的な位置 · 姿勢を計算することができます。

- 天井カメラの位置 · 姿勢のキャリブレーション

  キャリブレーションコンポーネント (LRFCalibration) はレーザレンジファインダのキャリブレーション用モジュールとして開発されましたが、2 次元平面上での位置を獲得するセンサであれば本コンポーネント群を同じように適用することが可能です。例えば、天井から真下を見下ろす天井カメラの位置 · 姿勢のキャリブレーションなどが考えられます。

4 謝辞

レーザレンジファインダコンポーネント (LRFComponent) の開発に当たりましては、東京大学生産技術研究所橋本研究室の川路浩平氏、Dražen Brščić 氏、佐々木毅が作成した C++ UrgLaser クラスのコードの一部を利用しております。川路浩平氏、Dražen Brščić 氏の両名に感謝申し上げます。