姿勢限定版物体位置姿勢推定コンポーネント

平成 24 年 2 月 22 日

豊橋技術科学大学　行動知能システム学研究室

1. このコンポーネントについて

このコンポーネントはスイス MESA 社製の 3 次元距離測定カメラ SwissRanger SR4000 からのデータを基に，机上に直立状態で置かれた複数個の直方体物体と円筒状物体の 3 次元空間上での位置と姿勢を推定し，その結果を共通形式で出力するコンポーネントである．

2. 開発・動作環境

このコンポーネントは以下の環境で開発し，動作確認をしている．

- Windows XP Pro SP3
- Open-rtm-aist 1.0.0(C++版)
- Microsoft Visual Studio 2008
- OpenCV 2.1

3. 入出力データポート

ポート名	データ型	入出力	備考
3DPointCloud	SRSensor::idl::TimedRange3DList	Inport	3 次元距離データの入力
SwissRangerData	SRSensor::idl::TimedSwissRanger	Inport	SwissRanger 固有のデータの入力
RecognitionResult	RTC::TimedDoubleSeq	Outport	認識結果の出力

4. サービスポート

ポート名	データ型	provider/ consumer	備考
MDService	ModelDefinitionRTC::idl::MDService	Consumer	位置姿勢推定を行う物体の形状データを取得
OPEService	ObjectPoseEstimationRTC::idl::OPEService	Provider	センサのキャリブレーション, 背景点群の取得を行う
ERVService	ObjectPoseEstimationRTC::idl::ERVService	Consumer	位置姿勢推定の結果を画像に表示する
RecognitionService	RecognitionService	Provider	指定された ID の物体の位置姿勢推定を行う

5. 各データ型・インターフェースについて

- SRSensor::idl::TimedRange3DList

SRSensor::idl::TimedRange3DList は株式会社セックが開発した SwissRanger SR4000 用のコンポーネントで使用されているデータ型である. 3 次元距離測定カメラから 3 次元距離データを取得するために用いられる.

メンバ名	データ型	備考
tm	RTC::Time	タイムスタンプ
data	SRSensor::idl::Point3DList	距離データ構造体

- SRsensor::idl::Point3DList

メンバ名	データ型	備考
range	SRSensor::idl::ChangedRange	sequence<RTC::Point3D>型として定義 3 次元距離データが 1 次元配列[25344]固定で格納
height	unsigned short	計測範囲の縦幅[144]固定
width	unsigned short	計測範囲の横幅[176]固定

- SRsensor::idl::ChangedRange

メンバ名	データ型	備考
x	double	カメラ前面の平面に対し，右手系の座標系において水平方向に左をプラスとした距離 単位:m
y	double	カメラ前面の平面に対し，右手系の座標系において垂直方向に上をプラスとした距離 単位:m
z	double	カメラ前面の平面に対し，右手系の座標系において光軸に沿ってカメラから離れる方向をプラスとした距離 単位:m

- SRSensor::idl::TimedSwissRanger

SRSensor::idl::TimedSwissRanger は株式会社セックが開発した SwissRanger SR4000 用のコンポーネントで使用されているデータ型である. SwissaRanger 固有のデータを取得するために用いられる.

メンバ名	データ型	備考
tm	RTC::Time	タイムスタンプ
data	SRSensor::idl::SwissRanger	SwissRanger 固有データ構造体

- SRSensor::idl::SwissRanger

メンバ名	データ型	備考
ittime	unsigned short	インテグレーションタイム（不使用）
confidence	Map	距離，明暗データとその瞬間のそれらの変動から求まるデータの信頼性 値の範囲：0～65535 単位：N/A （不使用）
amplitude	Map	反射強度 値の範囲：0～8191 単位：N/A

distance	rangeMap	距離データ 単位：m （不使用）

なお，Map 型は typedef long Map[144][176],
rangeMap 型は typedef double rangeMap[144][176]として定義している．

- RecognitionResult

位置姿勢推定の結果を表す．これは共通のインターフェース仕様に準拠しており，(20*n)個の実数型のデータ列からなる．n は推定された物体の個数である．

物体 1 個あたりのデータの内訳を以下に示す．

要素数	内容	備考
0	カメラセット ID	未使用
1	モデル ID	認識に用いたモデルの ID
2	認識候補番号	未使用
3	座標系番号	未使用
4	評価値	未使用
5	エラーコード	エラーコード（後述）
6	予備 1	未定義
7	予備 2	未定義
8	R_{00}	
9	R_{01}	
10	R_{02}	
11	T_x	
12	R_{10}	
13	R_{11}	
14	R_{12}	
15	T_y	
16	R_{20}	
17	R_{21}	
18	R_{22}	
19	T_z	
⋮	⋮	⋮

なおRとTはモデルから物体への回転と移動を表す座標変換行列の要素を意味する．モデルはセンサ座標系原点を中心としてつくられる．　角度の単位は[rad]，距離の単位は[mm]である．

$$\begin{pmatrix} R_{00} & R_{01} & R_{02} & T_x \\ R_{10} & R_{11} & R_{12} & T_y \\ R_{20} & R_{21} & R_{22} & T_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

エラーコードを以下に示す.

番号	意味
-3	ファイルのオープンに失敗した
-102	推定に失敗した
-105	指定された ID のモデルがデータベース内に見つからない

- ModelDefinitionRTC::idl::MDService

ModelDefinitionRTC::idl::MDService は位置姿勢推定を行う物体の形状データを取得するためのインターフェースである.

サービス名	引数	戻り値	備考
getObjectModel	long id	ModelDefinitionRTC::idl::ObjectModel	指定された ID のモデルを取得
getModelIDList	void	ModelDefinitionRTC::idl::ModelIDList	管理しているモデル ID のリストを取得

- ModelDefinitionRTC::idl::ObjectModel

ModelDefinitionRTC::idl::ObjectModel は，物体の形状を表すデータ型である． メンバ Model_category が 0 のときは直方体を，1 のときは円筒をそれぞれ表す.

メンバ名	データ型	備考
Model_id	short	モデル ID
Model_category	short	モデルの種類（0：直方体，1：円筒）
wx	float	直方体モデルの幅（x 軸方向長さ）の半分
wy	float	直方体モデルの奥行き（y 軸方向長さ）の半分
wz	float	直方体モデルの高さ（z 軸方向長さ）の半分
height	float	円筒モデルの高さの半分
radius	float	円筒モデルの上面・底面の半径

なお，使用しないメンバにはそれぞれ無効値 0.0 が入る.

- ModelDefinitionRTC::idl::ModelIDList

メンバ名	データ型	備考
idlist	sequence <long>	管理しているモデル ID のリスト

- ObjectPoseEstimationRTC::idl::OPEService

ObjectPoseEstimationRTC::idl::OPEService は物体の位置姿勢推定のための各機能を提供するインターフェースである.

calibrationSensor は, SwissRangerData から入力されたチェスパターンの反射強度画像を用い，OpenCV のカメラキャリブレーション関数によって 3 次元距離測定カメラの外部パラメータを求める.

キャリブレーションの際には，机の上にあらかじめ世界座標系を定め，図 1 のような OpenCV のキャリブレーションパターンを定めた座標系に合わせて置く．キャリブレーションが成功すると，カメラ座標系の原点から世界座標系の原点に座標変換を行う回転行列が得られる．世界座標系の原点は，カメラから向かってキャリブレーションパターンの左上のコーナーを原点とし，原点から右側の方向に Y 軸，下側の方向に X 軸，鉛直方向に Z 軸をとる（図 2）.

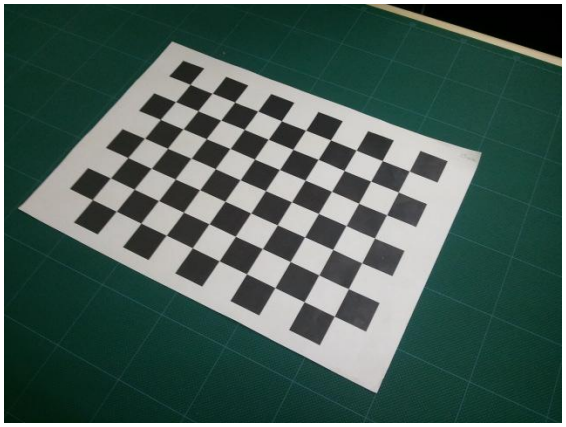

図 1　キャリブレーションパターン

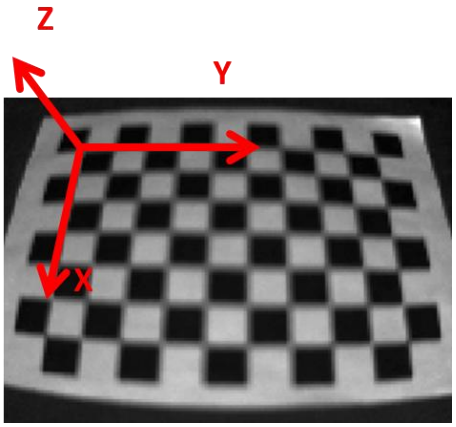

図 2　世界座標系

また，本コンポーネントで物体位置姿勢推定を行う際に，机平面の方程式を計算し，その平面と取得した 3 次元点との距離の差と，取得した 3 次元点ごとに周囲の点で平面を構成し，その平面の法線と机平面の法線との向きの違いを計算することにより，物体を示す 3 次元点群を取得している． getBackground は，机平面の方程式の算出のための背景の 3 次元点群を取得する．

カメラのキャリブレーションを行わず，SwissRanger の座標系上で物体の位置姿勢を取得する際は，getBackground のみを使用する．

getObjectPose は，OpenCV の世界座標系上での物体の位置，各軸周りの姿勢の計 6 自由度のパラメータと，その物体の ID を要素とする可変長のリストを取得する．なお，推定結果の単位は位置が[mm]，姿勢が[rad]である．（本コンポーネントでは使用しない）

サービス名	引数	戻り値	備考
calibrationSensor	なし	boolean	3 次元距離測定カメラのキャリブレーション
getBackground	なし	boolean	背景の 3 次元点群の取得と机平面の方程式の算出
getObjectPose	long id(不使用)	ObjectPoseEstimationRTC::idl::Pose3DList	推定した物体の ID と 3 次元空間上での位置姿勢のリストを取得（不使用）

- ObjectPoseEstimationRTC::idl::ERVService

ObjectPoseEstimationRTC::idl::ERVService は，距離画像上にモデルのワイヤーフレームを出力することにより物体位置姿勢推定の結果を表示するためのインターフェースである．

サービス名	引数	戻り値	備考
initializeSuperImpose	なし	boolean	結果を表示するための距離画像を生成
showVertices	sequence <RTC::Point3D>	boolean	各頂点の 3 次元座標を基に結果を表示する

- RecognitionService

RecognitionService は，指定した ID の物体の位置姿勢推定を行うためのインターフェースである．共通のインターフェース仕様に準拠している．

サービス名	引数	戻り値	備考
setModelID	long ModelID	void	指定したモデル ID の位置姿勢推定を開始する 認識が成功すると，OutPort から認識結果を出力する

データベースにないモデル ID を指定すると，見つかった物体全ての位置姿勢推定を行う．このときは，各点群に対してデータベースが管理している全ての物体データを当てはめ，それぞれ最も当てはまりがよかった推定結果のリストを出力する．

6. コンフィギュレーション

変数名	データ型	備考
Chess_size	double	キャリブレーションパターンの 1 マスのサイズ 単位：mm
DistanceThresh	double	背景点群の取得と，物体の姿勢推定の際に使用する距離データの奥行きの上限値　単位：m
Pat_col	int	キャリブレーションパターンの列数
Pat_row	int	キャリブレーションパターンの行数

7. 準備

7.1. ライブラリ・ドライバのインストール

このコンポーネントを使用するためには，OpenCV ライブラリが必要である．インストール方法を以下に示す．

a）配布サイト（http://sourceforge.net/projects/opencvlibrary/files/ ）などで OpenCV-2.1.0-win32-vs2008 をダウンロードし，実行する．

b）環境変数 Path に”C:¥OpenCV2.1¥bin”（cv210.dll，cxcore210.dll があるフォルダ）を追加する．このアドレスは標準設定でインストールした場合である．

上記以外に新しく環境変数を設定する必要はない．

また， SwissRanger SR4000 を使用するために，その付属のデバイスドライバをインストールする．

なお，本コンポーネントは単体で動作させることが出来ないため，3DPointCloud，SwissRangerData につなぐためのコンポーネントである株式会社セック製の SwissRangerRTC を用意する必要がある．

7.2. ビルド

a) 上記のライブラリをインストールしたのち, <コンポーネント名>_vc9.sln を開き, Visual Studio のメニューバーより，「ツール＞オプション」を選択し，「オプション」ウィンドウを表示させる(図 3).

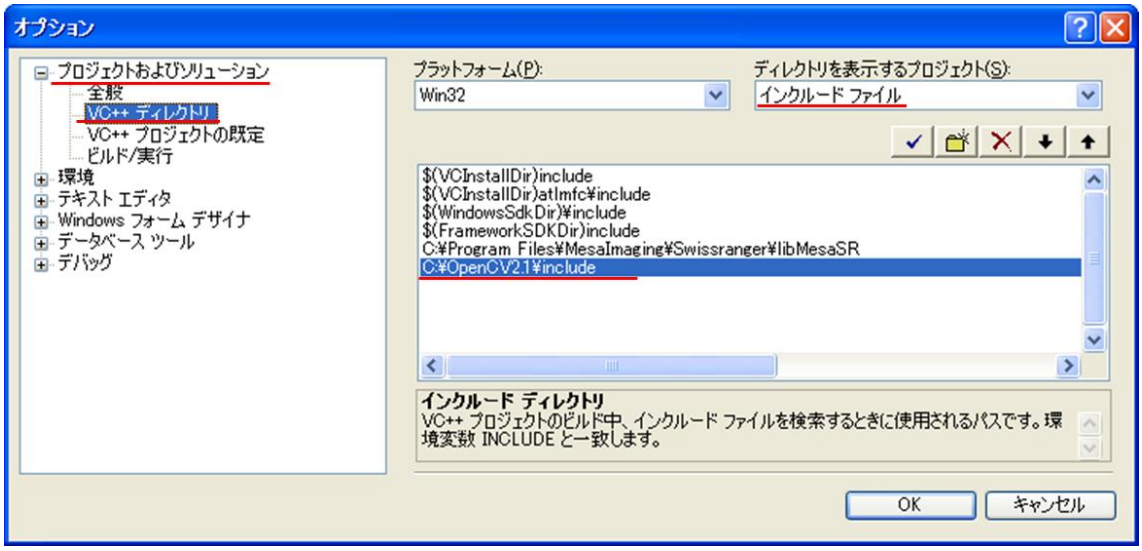

図 3 「オプション」ウィンドウの表示

b)「プロジェクトおよびソリューション」のメニューから，「VC++ディレクトリ」を選択し，「インクルードファイル」と「ライブラリファイル」の欄にそれぞれ OpenCV2.1 の include フォルダと lib フォルダのパスを記述する．

c) 次に，「ソリューションエクスプローラー」において ObjectPoseEstimationComp を選択したのち，メニューバーより「プロジェクト＞プロパティ」を選択し，「ObjectPoseEstimationComp プロパティページ」ウィンドウを表示させる(図 4).

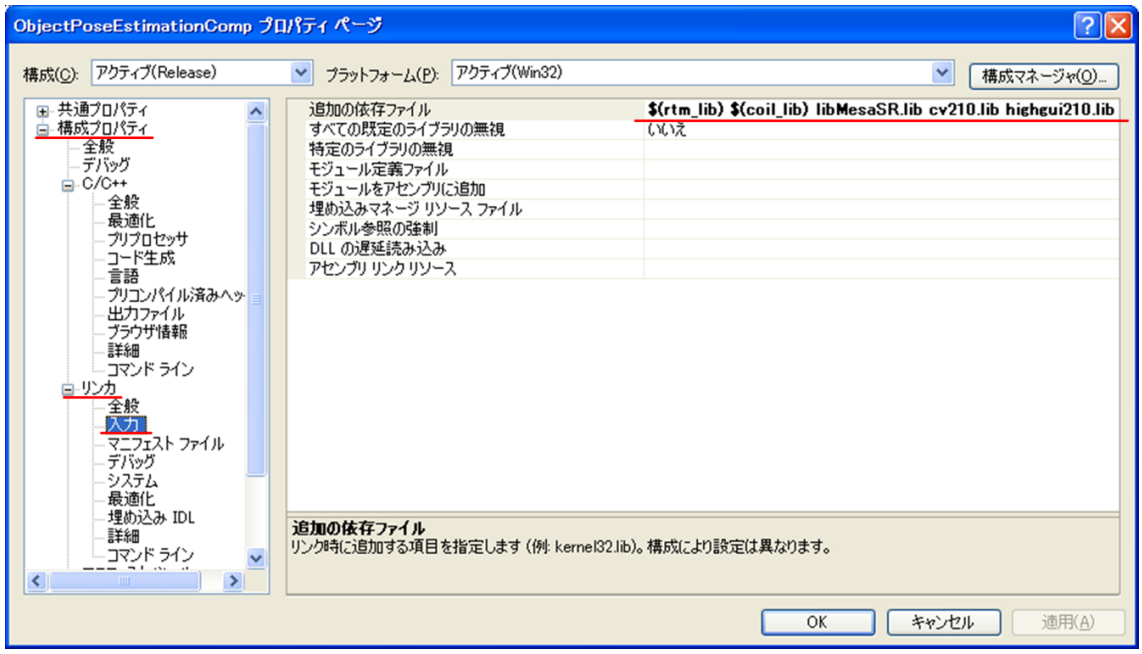

図 4 「ObjectPoseEstimationComp プロパティページ」ウィンドウの表示

d）ツリーメニューから「構成プロパティ＞リンカ＞入力」を選択し，「追加の依存ファイル」の欄に”libMesaSR.lib”, “cv210.lib”, “highgui210.lib”, “cxcore210.lib”の各ファイルを追加する．

e）Release モードに設定後，F7 キーを押してビルドを行う．

8. 実行手順

8.1 圧縮ファイルの展開

ファイルの中身は図 5 のようになっている．

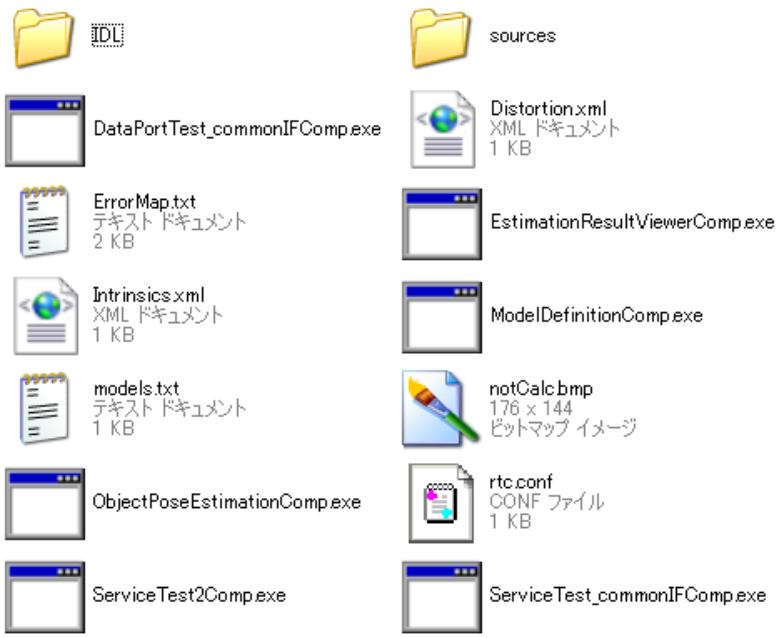

図 5　解凍フォルダのファイル構成

a）”IDL”フォルダ

このコンポーネントが使用しているデータ型，サービスを定義した IDL ファイルが格納されているフォルダ．

b）”sources”フォルダ

このコンポーネントのソースとプロジェクトファイルが格納されているフォルダ．

c）rtc.conf

RT コンポーネントの各種設定（ネーミングサービス・動作周波数）を記述するための設定ファイル.

d） Distortion.xml

3 次元距離測定カメラの歪み係数が記述されているファイル. センサキャリブレーションサービスの実行時，推定結果のワイヤーフレーム画像表示に使用する.

e）Intrinsics.xml

3 次元距離測定カメラの内部パラメータが記述されているファイル. センサキャリブレーションサービスの実行時，推定結果のワイヤーフレーム画像表示に使用する.

f）ErrorMap.txt

世界座標系上での推定結果のずれを修正するために，センサ座標系上での位置と，そこにおける世界座標系上での誤差ベクトルを記述したファイル.

本コンポーネントでは反射強度画像を使用してキャリブレーションを行っているが，SR-4000 では距離計測系と撮像素子とが分かれているため，そこからずれが生じてしまう．それを修正するために，推定を行う範囲で上記の要素からなる誤差マップをつくり，推定結果に適用する.

誤差マップは，世界座標系原点からある物体の重心までの距離と，その物体の世界座標系上での位置推定結果との誤差を調べることにより作成される. 物体は，幅 40mm，奥行き 30mm，高さ 50mm の直方体を用いた．その物体を世界座標原点から x 軸方向に 50mm，y 軸方向に 50mm 間隔に計 35 箇所に配置し，各地点における誤差を測りファイルに記述した．これにより，SwissRanger が机から約 500mm 離れているときの視野範囲全域に対応する．なお，机に対するセンサの高さによっては，誤差マップを適用しても推定結果の誤差が大きくなってしまうことがある．この誤差は世界座標系の z 軸方向に顕著にみられる.

推定時には，結果のセンサ座標系における位置パラメータと 2 次元距離が最も近い地点で観測された誤差ベクトルを適用する.

記述は，以下のフォーマットに従う.

・誤差ベクトルの x 座標，y 座標，z 座標，その誤差が観測されたセンサ座標系上の x 座標，y 座標を記述．各データはカンマで区切られる．単位は[mm].

g）ObjectPoseEstimationComp.exe

本コンポーネントの実行ファイル．Distortion.xml, Intrinsics.xml, ErrorMap.txt をこれと同一のフォルダに置く必要がある．

h）ModelDefinitionComp.exe

本コンポーネントの動作テスト用に作成したモデルデータベース RTC の実行ファイル． ModelDefinitionRTC::idl::MDService のサービスを提供する．モデルデータはテキストファイルとして管理されている．記述は以下のフォーマットに従う．

①直方体データ

・行の先頭に”B”.

・スペースで区切り，モデル ID，幅の半分（x 軸方向），奥行きの半分（y 軸方向），高さの半分（z 軸方向）を記述．単位は[m].

記述例： ID 0 番，幅 40mm，奥行き 30mm，高さ 50mm の直方体

```
B 0 0.02 0.015 0.025
```

②円筒データ

・行の先頭に”C”.

・スペースで区切り，モデル ID，半径，高さの半分を記述．単位は[m].

記述例： ID1 番，半径 15mm，高さ 60mm の円筒

```
C 1 0.015 0.03
```

コンフィギュレーションとして FileName という項目があり，モデルデータを記述したテキストファイルのパスを指定する．デフォルトでは”models.txt”となっている．

i）models.txt

本コンポーネントの動作テスト用に作成した，モデルデータを記述したテキストファイル．

j）notCalc.bmp

SwissRanger の計測範囲の中で，計算に使用する領域を RGB(255, 255, 255)で表した 176×144 のマスク画像．使用しない領域は RGB(0, 0, 0)で表す．デフォルトではすべての画素が RGB(255, 255, 255)となっている．

k）ServiceTest2Comp.exe

本コンポーネントの動作テスト用に作成したキャリブレーションサービス・背景取得サービスのテスト用 RTC の実行ファイル．コンソール画面でのコマンド入力により ObjectPoseEstimationRTC::idl::OPEService の各サービスを呼び出す．コマンドは，calibrationSensor が'c'，getBackground が'b'となっている．

l）EstimationResultViewerComp.exe

本コンポーネントの動作テスト用に作成した位置姿勢推定結果表示用 RTC の実行ファイル．EstimationResultViewerRTC::idl::ERVService のサービスを提供する．コンフィギュレーションとして SAVE_IMAGE という項目があり，ここに 1 を設定すると出力した画像を実行ファイルと同一のフォルダに保存する．また， Distorton.xml, Intrinsics.xml を実行ファイルと同一のフォルダに置く必要がある．

m）ServiceTest_commonIFComp.exe

本コンポーネントの動作テスト用に作成した位置姿勢推定サービスのテスト用 RTC の実行ファイル．コンソール画面でのコマンド入力により RecognitionService の setModelID サービスを呼び出す．コマンドは，認識したい物体の ID となっている．

n）DataPortTest_commonIFComp.exe

本コンポーネントの動作テスト用に作成した，データポート出力のテスト用 RTC の実行ファイル. コンソール画面に本コンポーネントが出力した推定結果が表示される (図 6)．

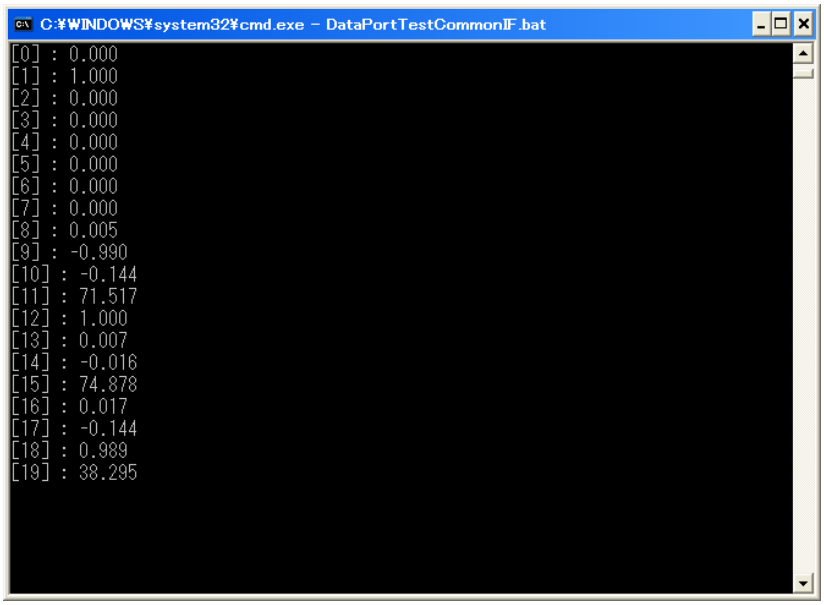

図 6　DataPortTest_commonIF のコンソール画面における，位置姿勢推定結果の出力例

8.2 ネームサーバーの起動

スタート＞すべてのプログラム＞OpenRTM-aist＞C++＞examples＞Start Naming Service を選択する.

8.3 モジュールの起動

展開したフォルダ内にある”ObjectPoseEstimationComp.exe”を起動する. また, 3DPointCloud, SwissRangerData につなぐための SwissRangerRTC, MDService につなぐためのモデルデータベース用 RTC, RecognitionResult につなぐためのデータポート出力テスト用 RTC, ERVService につなぐための位置姿勢推定結果表示用 RTC, OPEService につなぐためのキャリブレーションサービス・背景取得サービステスト用 RTC, RecognitionService につなぐための位置姿勢推定サービステスト用 RTC も起動しておく.

8.4 センサデバイスの接続

SwissRangerSR4000 を Ethernet インターフェースもしくは, URB2.0 インターフェースを PC に接続する.

8.5 RTC System Editor での操作

eclipse を起動する.

a) メニューバーより,「ウィンドウ＞パースペクティブを開く＞その他」を選択し,「パースペクティブを開く」ウィンドウを表示させる(図 7).

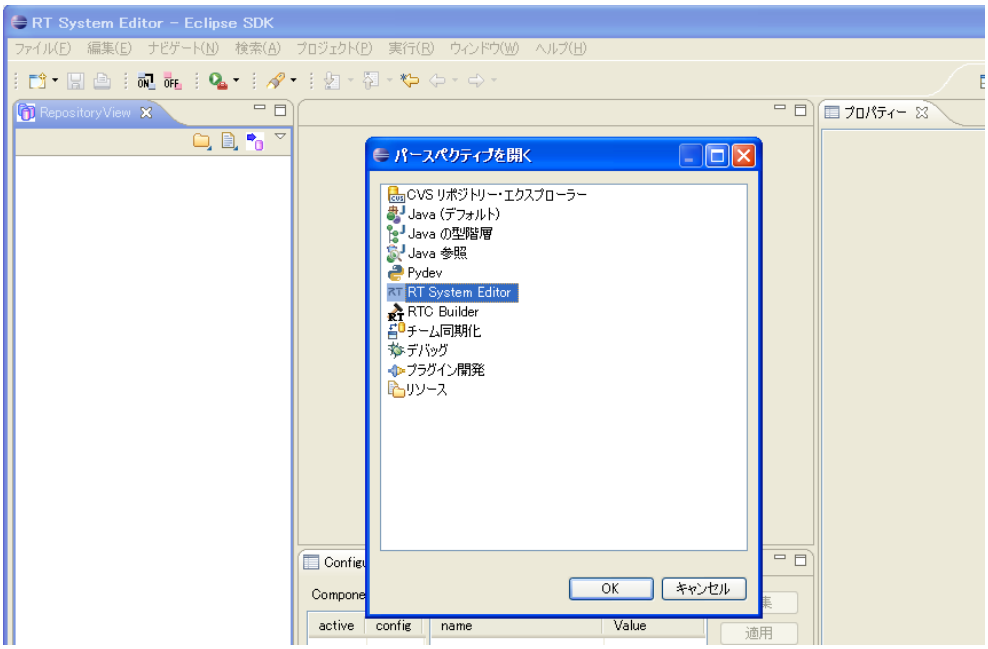

図 7 「パースペクティブを開く」ウィンドウの表示

b)「RT System Editor」を選択し，OK ボタンを押す.

c) 図 8 に示す赤い丸で囲まれたアイコンを選択する．NameServiceView が表示されていない場合は，メニューバー＞ウィンドウ＞ビューの表示＞NameServiceView を選択する.

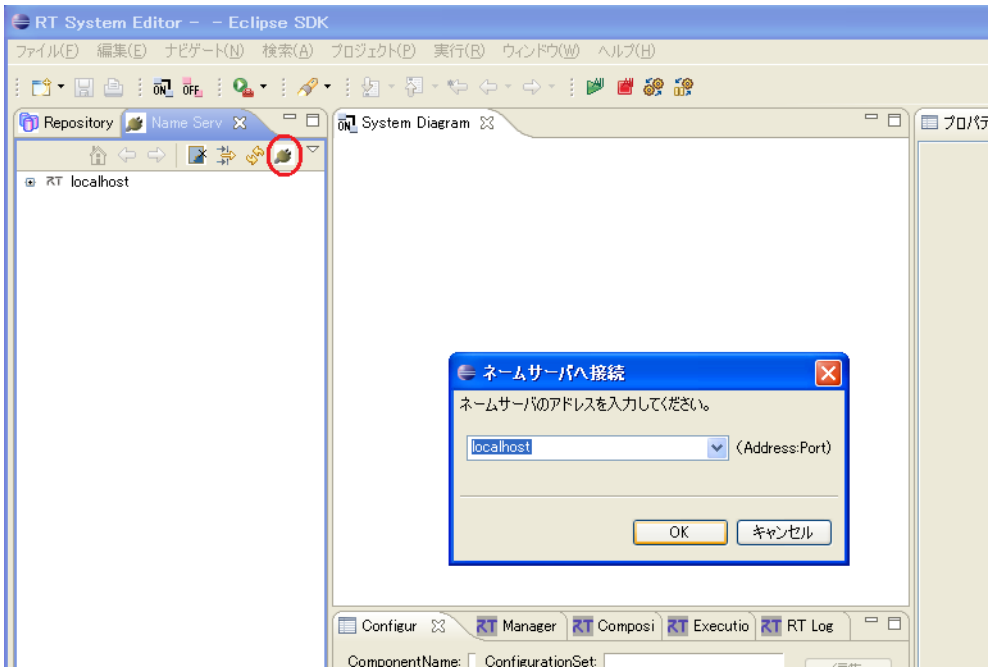

図 8　NameServiceView の操作

d）図 8 のように Connect Name Server の Address Port に「localhost」と入力して OK を選択する．

e）NameServiceView の localhost のツリーを開き，起動したモジュール群が表示されていることを確認する．

f）メニューバーより，ファイル＞Open New System Editor を選択する．

g）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．

h）図 9 に示すようにモジュールを接続する．

i）全てのコンポーネントをアクティベートする．

9. 使用例

以下に本コンポーネントの接続例を示す．図中の SwissRanger4k は，株式会社セック製 SwissRanger 用コンポーネントである．また，ModelDefinition はモデルデータベース RTC，DataPortTest_commonIF はデータポート出力テスト用 RTC，ServiceTest2 はキャリブレーションサービス・背景取得サービステスト用 RTC，ServiceTest_commonIF は位置姿勢推定サービステスト用 RTC，EstimationResultViewer は位置姿勢推定結果表示用 RTC である．

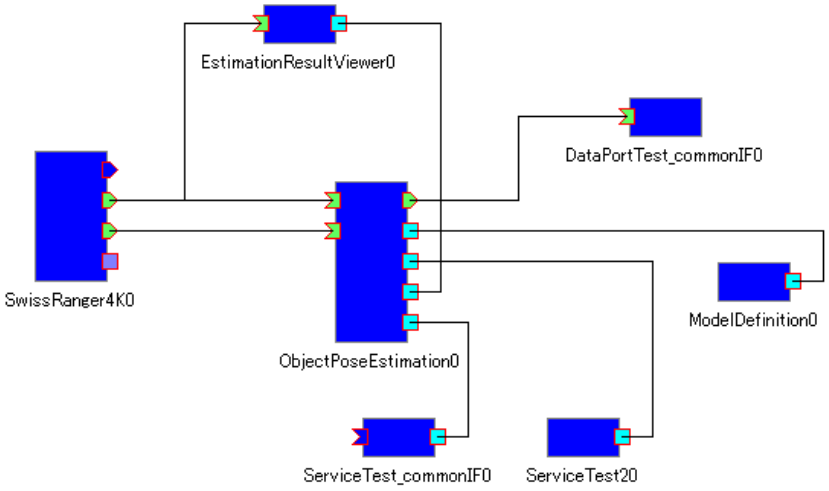

図 9　RT コンポーネントの接続例

まず，位置姿勢推定サービスである setModelID を使用する前に，3 次元距離測定カメラの外部パラメータをセンサキャリブレーションサービスである calibrationSensor によって求める必要がある．calibrationSensor が成功すると，サービスが戻り値 true を返し，図 10 のようなウィンドウが表示される．

図 10　キャリブレーションの成功

キャリブレーションが成功したら，背景の 3 次元点群を取得するために getBackground を呼ぶ．それから，setModelID により物体の世界座標系での位置姿勢推定が可能となる．

カメラのキャリブレーションを行わず，SwissRanger 座標系上のみにおいて物体の位置姿勢を推定する際は，getBackground のみを呼んだのち，setModelID を呼ぶ．

なお，　calibrationSensor と getBackground の実行時に，座標変換に使用するパラメータを格納するファイルである”Translation.xml”と，”Rotation.xml”，物体点群の取り出しに使用する机平面の方程式のパラメータを格納するファイルである”planeparameter.txt”が

保存される．RTC 内部での処理ではそれらのファイルが参照されるため，カメラと机の位置関係が変わらない限り，calibrationSensor と getBackground は一度だけ呼べばよい．

位置姿勢推定が成功すると，DataPortTest_commonIF のコンソール画面に図 6 に示すような位置姿勢パラメータが表示され，EstimationResultViewer が図 11 に示すような推定結果の画像を表示する．

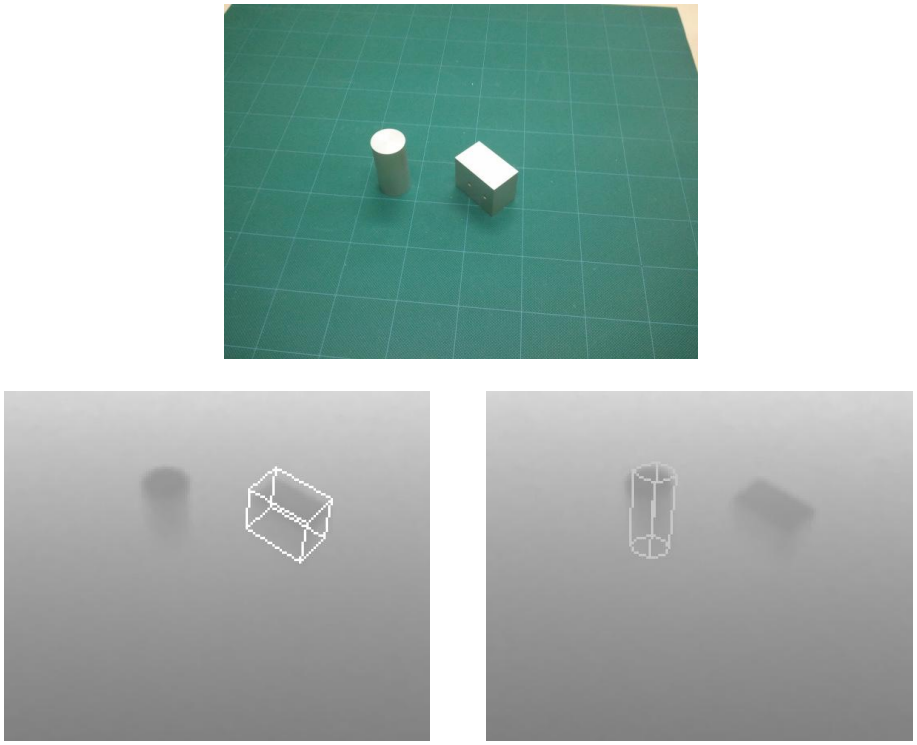

図 11　実験環境の様子と，それぞれの推定結果をワイヤーフレームとして出力した画像

10. 連絡先について

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．

11. ライセンス

本コンポーネントは修正 BSD ライセンスを適用している．