OpenVGR

操作手順書

Ver.0.9.0

2011年12月21日

独立行政法人 産業技術総合研究所

本文書の取り扱いについて

○ 本書に掲載する情報は、使用者に有用なものであるように万全を期していますが、内容の正確性、最新性、その他一切の事項について保証をするものではありません。

○ 本書の著者および著作権者は、使用者がこの文書から得たまたは得られなかった情報から生じる損害に対しても、一切責任を負いません。

○ 本書は使用者への事前の予告なしに変更、削除、公開の中止を行うことがあります。

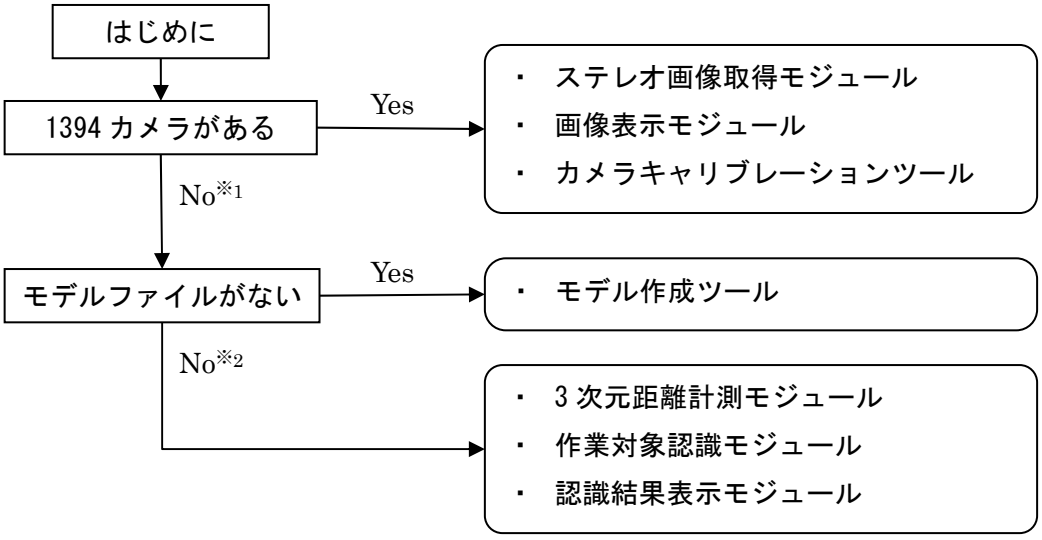
【連絡先】
（独）産業技術総合研究所 知能システム研究部門 タスクビジョン研究グループ
〒305-8568 茨城県つくば市梅園 1-1-1 中央第二
E-Mail: openvgr-contact@m.aist.go.jp

目次

1. はじめに

1.1. 本書の読み方

本書では、使用者が作業対象認識モジュール群を実際に動かすために必要な操作を段階的に説明します。システムを導入する環境に合わせて、それぞれのモジュールの説明を読み進めてください。



※ 1 ステレオ画像とキャリブレーションデータが必要

※ 2 すでにモデルを作成ずみの場合

2. 導入方法

2.1. 動作環境

本プログラムは以下の環境で動作します。

使用言語：C++，C 言語

動作環境：

PC：CPU Core2Duo 以上，メモリ 1GB 以上，ハードディスク 10GB 以上

OS：Ubuntu 10.04 LTS Desktop 日本語 Remix （x86 32bit 環境）

カメラを使用する場合：IEEE 1394b カメラ （IIDC 1.31 準拠）

Point Grey Research 社製 Flea2 カメラで動作確認

開発環境：

OpenRTM-aist 1.0.0-RELEASE C++版

OpenRTM Eclipse tools 1.0-RELEASE

GNU Compiler Collection 4.4.3

OpenCV 2.0

libdc1394-2

2.2. インストール

2.2.1. パッケージのインストール,

作業対象認識モジュール群を動作させるために、以下のソフトウェアパッケージをインストールします。

- OpenRTM-aist 1.0.0 RELEASE C++ 版
- OpenRTM Eclipse tools 1.0-RELEASE
- OpenCV 2.0

公式ページ[1]からOpenRTM-aist（C++ 版）1.0.0-RELEASEとOpenRTM Eclipse tools 1.0-RELEASEをダウンロードし、パッケージのインストール及び設定を以下の手順に従ってください。

■OpenRTM-aist 1.0.0 RELEASE C++ 版のインストールと設定

OpenRTM-aistの公式サイトにOpenRTM-aist（C++ 版）1.0.0-RELEASEの一括ダウンロード用シェルスクリプトが用意されていますので、webページの説明に従ってインストールしてください。（詳細なインストール方法については「5.1OpenRTM-aistのインストール方法」を参照してください。）

図 1　公式サイト Linux パッケージの Web ページ（2011 年 6 月 22 日現在）

パッケージ名	OpenRTM-aist (C++ 版) 1.0.0-RELEASE
公式サイト内ページ	http://www.openrtm.org/openrtm/ja/content/openrtm-aist-100-release#toc3
ダウンロードするファイル	pkg_install100_ubuntu.sh (http://openrtp.jp/openrtm/svn/OpenRTM-aist/trunk/OpenRTM-aist/build/pkg_install100_ubuntu.sh)

[1] http://www.openrtm.org/

■OpenRTM Eclipse tools 1.0-RELEASE のインストールと設定

次に OpenRTM Eclipse tools 1.0-RELEASE をインストールします。OpenRTM Eclipse tools 1.0-RELEASE とは、RT コンポーネントを扱う RTCBuilder および RTSystemEditor が組み込まれた Eclipse 統合開発環境です。OpenRTM-aist の公式サイトで配布されています。

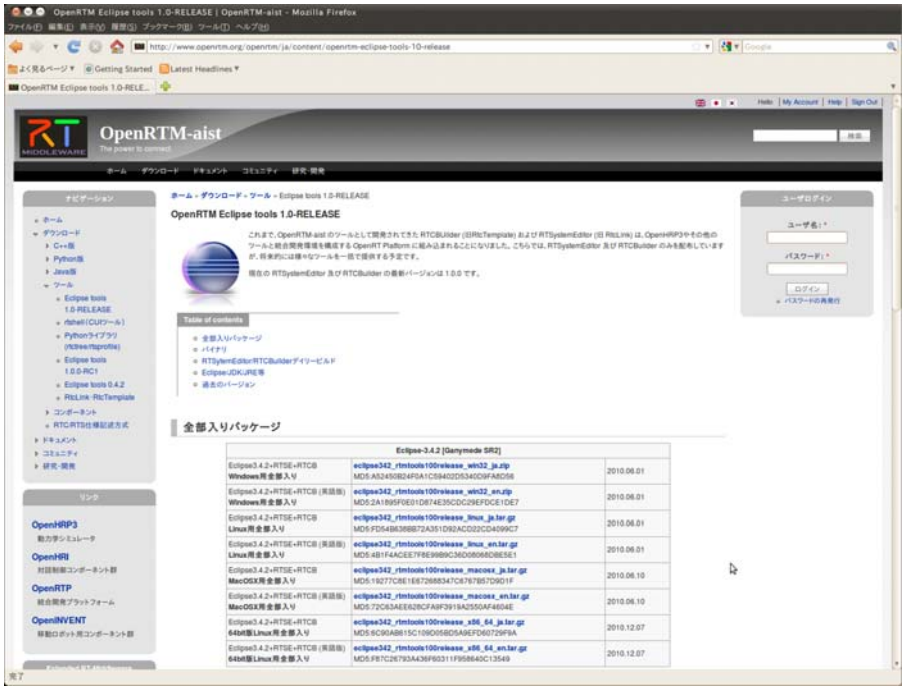


図 2　公式サイトの Eclipse tools の Web ページ（2011 年 6 月 22 日現在）

パッケージ名	Eclipse3.4.2+RTSE+RTCB Linux 用全部入り
公式サイト内ページ	http://www.openrtm.org/openrtm/ja/content/openrtm-eclipse-tools-10-release#toc0
ダウンロードするファイル	eclipse342_rtmtools100release_linux_ja.tar.gz (http://www.openrtm.org/pub/OpenRTM-aist/tools/1.0.0/eclipse342_rtmtools100release_linux_ja.tar.gz)

eclipse342_rtmtools100release_linux_ja.tar.gz をダウンロードして展開すると、eclipse というディレクトリができます。

Eclipseを動かすために、JavaのJRE (Java Runtime Environment)またはJDK (Java Development Kit)が必要になります。http://www.java.com/ja/download/ からJREのLinux自己解凍ファイルをダウンロードしてください。そのファイルを展開してできるディレク

トリをjreという名前でeclipseディレクトリに移動してください。
（詳細なインストール方法については「5.3Ubuntu10.04 環境へのインストールの注意事項」を参照してください。）

2.2.2. 追加パッケージのインストール

■OpenCV のインストール

次に OpenCV 2.0 パッケージをインストールします。OpenCV 2.0 パッケージは、画像処理のためのソフトウェアパッケージです。Synaptic パッケージマネージャを起動し、以下のソフトウェアパッケージをインストールしてください。なお OpenCV 2.1 以降には対応していません（2011 年 6 月現在）。

- libcv4
- libcvaux4
- libhighgui4
- libcv-dev
- libcvaux-dev
- libhighgui-dev

図 3　synaptic の画面

■libdc1394-2 のインストール

同様に Synaptic パッケージマネージャで以下のソフトウェアパッケージをインストールしてください。

- libdc1394-22-dev

2.2.3. RTコンポーネントをインストールする場所を決める

ディスクトップ上のツールバーの[アプリケーション]から[アクセサリ]を選択して[端末]をクリックし、コンソールを開いてください。(図 4)

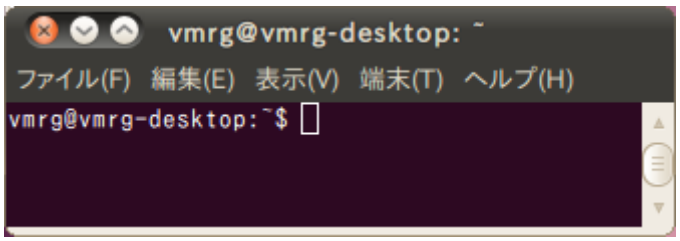


図 4 コンソール画面(ターミナルエミュレータ GNOME 端末)

ホームディレクトリの下に、作業対象認識モジュール群をインストールするディレクトリを新しく作成します。インストール先は自由に選ぶことができます。ここでは~/opt/OpenVGR ディレクトリにインストールするものとして説明します。

下のようにコマンドを実行して、~/opt ディレクトリを用意してください。

```
$ mkdir ~/opt
$ cd ~/opt
```

2.2.4. RTコンポーネントとツールをビルドする

古い OpenVGR ディレクトリが~/opt ディレクトリの下にあれば削除してください。~/opt ディレクトリの直下で `OpenVGR-`*`バージョン番号`*`.tgz` を展開します。すると `OpenVGR-`*`バージョン番号`*のディレクトリができあがるので、ディレクトリ名を OpenVGR に変更します。

```
$ rm -rf OpenVGR
$ tar xzvf OpenVGR-バージョン番号.tgz
$ mv OpenVGR-バージョン番号  OpenVGR
```

次に RT コンポーネントおよびツールをビルドします。OpenVGR ディレクトリに移動して make コマンドを実行します。するとプログラムのビルド過程がしばらく表示され続け、ビルドが終了すると再びコマンドプロンプトが表示されます。

```
$ cd OpenVGR
$ make
（ビルドの過程が表示される）
```

2.3. 準備

2.3.1. サンプルのコピー

次に作業用のディレクトリを作り、これから使用するモジュール（RT コンポーネントおよびツール）とサンプルの設定ファイルを用意します。作業を行うディレクトリの名前は自由ですが、ここではソースコードと一緒に展開される build ディレクトリを使います。

インストール作業からの続きです。build ディレクトリに移動して make を実行すると、各モジュールへのシンボリックリンクが作成されます。具体的な例を下に示します。

```
$ cd build
$ make
（リンクを作成する過程が表示される）
```

最初にカメラを使わずに作業対象認識モジュールの動作を確認するため、サンプルの画像データとカメラキャリブレーションデータを全て build ディレクトリの下にコピーします。

```
$ cp -R ../example/dataset/image ./
$ cp -R ../example/dataset/calib ./
```

作業対象認識モジュールが動作するためには、三つのテキストファイルが必要になります。それらを build ディレクトリに配置します。

一つめは、作業対象認識モジュールが認識に使用するモデルファイル名とモデル ID を組にした一覧のテキストファイルです。ここではファイル名を ModelList.txt とします。サンプルのモデルと ModelList.txt を、ディレクトリごと build の下にコピーしてください。

```
$ cp -R ../example/model ./
```

二つめは認識のための様々なパラメータを記述したテキストファイルです。サンプルのパラメータファイルを build の下にコピーしてください。

```
$ cp -R ../example/dataset/param ./
```

三つめは RT コンポーネントの設定ファイルです。これは次節で詳しく解説します。rtc.conf というファイル名で build ディレクトリにあらかじめ用意されています。

2.3.2. 必ず設定する項目

OpenVGR の RT コンポーネントを実行するときに必要な rtc.conf は build ディレクトリに用意されたものが利用できます。このファイルには次の一文が記述されています。

```
corba.args: -ORBgiopMaxMsgSize 10485760
```

この文の最後の数字部分が、データポートで通信できるデータのバイト単位で指定された最大サイズになります（上記例では 10MB)。モジュール間の通信を担う omniORB の標準設定では､データポートでやり取りできるデータのサイズは 2MB までとなっていますが、本プログラムで扱う画像のデータサイズはこれよりも大きくなります。そのため、この上限値を大きくする必要があります。

ご使用になるカメラの解像度、台数、カラー画像かモノクロ画像かで、必要なデータサイズは異なりますので、適宜算出して設定してください。距離計測データはステレオ画像に加えて多くの 3 次元点群を出力しますので、さらにカメラの画素数の 24 倍程度のデータサイズが必要になります。モジュールが正常動作しているように見えて、データが何も送信されてこない場合は設定値を大きくして下さい。

通常のカメラの場合、次の要領でデータサイズを求めることができます。カメラの画素数と台数を掛けて総画素数を計算し、カラーなら 1 画素を 3 バイト、グレーなら 1 画素を 1 バイトに換算してください。

作成した rtc.conf ファイルは実行する RT コンポーネントと同じディレクトリに置いてください。あるいは、RT コンポーネントのプログラムの起動オプションとして、下記のようにファイルパスを指定して使います。

```
$ ./（実行する RT コンポーネント） –f （ディレクトリのパス）/rtc.conf
```

2.3.3. 必要に応じて設定する項目

■ステレオカメラについて

作業対象認識モジュール群は、2 台または 3 台のカメラで構成されるステレオカメラが必要です。代表的な構成を図 5に示します。これは三脚と雲台を使ってカメラを配置した場合の例です。カメラはそれぞれ、被写体が画像の中心付近に映るように向きを調整します。カメラを 3 台使用すれば、2 台では有効に活用できない水平線（2 台のカメラの並びに水平な線分）も、3 次元情報を計測する有効な手がかりになります。

カメラの位置と方向を厳密に調整する必要はありません。ただしカメラをしっかりと固定しておく必要があります。カメラキャリブレーションを行なった後に、カメラ同士の位置関係がわずかでも変わってしまうと計測結果が悪化します。

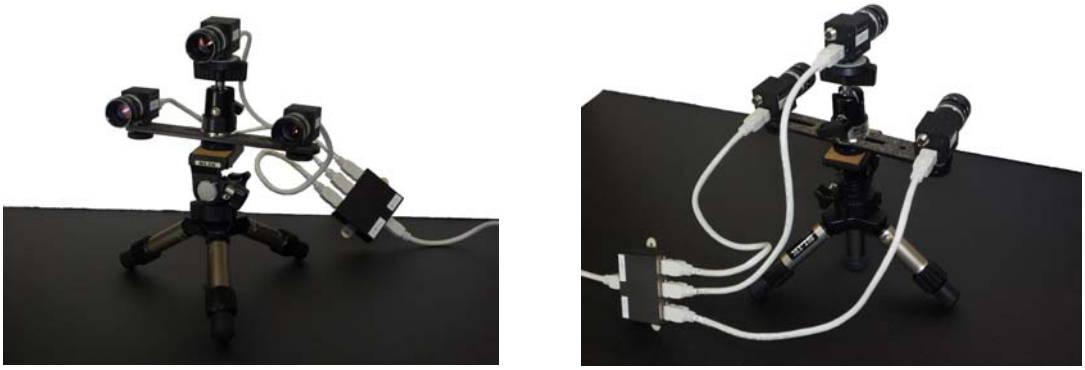


図 5　ステレオカメラの構成例

図 6　レンズ

図 7　レンズの取り付け方法

作業対象認識モジュール群がサポートするカメラは、IIDC1.31 仕様に準拠した非圧縮デジタルカメラ（以降、1394 カメラ）です。**図 5**のようにカメラを接続するには、4 本の 1394 ケーブル、ケーブルをつなぐ 1 台の 1394bハブ、PCに 1394bケーブルのコネクタを差し込むIEEE 1394bインターフェースカードが必要になります。

カメラを PC に接続し、正常に認識されると、システムのログにカメラの ID 番号が出力されます。ログは dmesg コマンドで見ることができます。

ログメッセージを表示する例を次に示します。最後の 3 行の GUID がカメラの ID 番号です。番号はカメラによって異なります。

```
$ dmesg | tail
[    13.797451] ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
[    24.013850] eth0: no IPv6 routers present
[15087.014221] ieee1394: The root node is not cycle master capable; selecting a new root node and resetting...
[15088.167012] ieee1394: Error parsing configrom for node 0-00:1023
[15088.845197] ieee1394: Error parsing configrom for node 0-02:1023
[15089.523410] ieee1394: Error parsing configrom for node 0-04:1023
[15089.523490] ieee1394: Node changed: 0-00:1023 -> 0-06:1023
[15093.407494] ieee1394: Node added: ID:BUS[0-00:1023]   GUID[00b09d01009bce7f]
[15093.409792] ieee1394: Node added: ID:BUS[0-02:1023]   GUID[00b09d01009bce7b]
[15093.413915] ieee1394: Node added: ID:BUS[0-04:1023]   GUID[00b09d01009bce76]
```

この GUID の表示の有無でカメラが認識されているかどうかを確認できます。

カメラが PC に正常に認識されることを確認できたら、libraw1394 が起動してすぐ使えるように設定します。libraw1394 は 1394 カメラを扱う RT コンポーネントに必要とされるライブラリです。PC の起動時に IEEE 1394 関連のデバイスファイルが自動的に用意され、常にカメラが使用できるようします。まず、設定ファイル /etc/udev/rules.d/50-raw1394.rules を作成します。vi または nano などのテキストエディタによって新しいテキストファイルを開いてください。

```
$ sudo vi /etc/udev/rules.d/50-raw1394.rules
[sudo] password for (ユーザ名):
```

下記の内容をファイルに記載してください。

```
KERNEL=="raw1394", GROUP="root", MODE="0666"
```

Linux システム全体に対する設定項目を含みますので、記述に間違いが無いよう十分に気を付けてください。以上で libraw1394 の設定は終わりです。ただし、システムの設定によってはカメラを使用するユーザを video グループへ追加する必要がありますので、ご注意ください。最後に設定を反映させるため、PC を再起動してください。

これで、1394 カメラを Ubuntu で利用するための準備ができました。カメラのキャリブレーションを行った後で、接続したカメラから画像取得モジュールによって画像を撮影することができるようになります。

■ieee1394board.0 について

ieee1394board.0 設定ファイルは、IEEE 1394 のインターフェースボードに接続されたカメラの設定値を記録します。プログラムがカメラから画像を撮影するとき、常に同じ状態を再現できるよう、あらかじめカメラの設定値を記録しておくためのものです。

各設定値の意味については，1394 カメラの標準仕様
IIDC 1394-based digital camera specification ver. 1.31
http://damien.douxchamps.net/ieee1394/libdc1394/iidc/
をご参照ください.

ieee1394board.0 の作成方法：

プログラム genconf を実行すると、接続されたカメラの ID 番号（0x で始まる 16 桁の英数字）を記載した ieee1394board.0 のひな形が生成されます。

```
$ ./genconf > ieee1394board.0
```

ieee1394board.0 ファイルの表記内容は下記のようになります。もしカメラが認識されていなかった場合、＜カメラの台数 n＞の数値は 0 となるのでカメラの接続からやり直してください。

```
0 <カメラの台数 n>
<カメラ ID 番号 1> 640x480-Y(mono) <Trigger 値>fps <カメラコントロール名 1> <値> <カメラコントロール名 2> <値> ・・・
                              :
                              :
<カメラ ID 番号 n> 640x480-Y(mono) <Trigger 値>fps <カメラコントロール名 1> <値> <カメラコントロール名 2> <値> ・・・
```

生成されたこの設定ファイルに必要な情報を与え、接続されたカメラの ieee1394board.0 を作成します。

まず、カメラが適度な明るさで被写体を撮影できるようカメラの設定値を調節します。設定値はカメラコントロール用のソフトである Coriander を使って調べていきます。

Coriander の操作方法:

あらかじめ Synaptic で Coriander をインストールしておいてください。

次のように入力して、Coriander をコンソールから起動します。

```
$ coriander &
```

Corianderを起動すると図 8のようにCameraタブが開かれた状態で表示されます。

図 8　Coriander の Camera 画面

CameraタブのCamera Selectでコントロールするカメラを選択できます（図 8）。Camera Selectのリストに並ぶカメラは順不同です。その下方にあるCamera informationの中のGUIDがカメラのID番号です（図 8）。このGUIDでカメラを識別します。

もし Coriander の操作中にカメラの映像が停まる／カメラが操作できないなどの症状が現れた場合には、一旦 Coriander を終了して、ieee1394 ケーブルを PC から抜き差しした後に再度 Coriander を起動してください。

次に現在選択中のカメラの画像を表示させます。Services タブを開いてください。

図 9　Coriander の Services の画面

Service内のReceiveタブを開いた状態で、その中央にあるTriggerの値に注目してください。複数のカメラを扱うので、一度に処理する画像の枚数を考慮する必要があります。ここではフレームレートを低めに設定して 7.5fpsにします（図 9)。

次にService内の一番上に並ぶボタンの中にあるDisplayボタンを押すと、ビューア画面が現れ、カメラに写る映像が表示されます（図 9)。

これからの作業はカメラの映像を見ながら行います。なお、Receive タブ内の Trigger の左横には ISO Control があり START / STOP、RESTART ボタンがありますが、STOP は押さないでください。静止画の状態ではカメラコントロール値の変更がビューアの表示に反映されなくなります。

次に Controls タブを開いて、取得するカメラコントロールの各値を調節します。

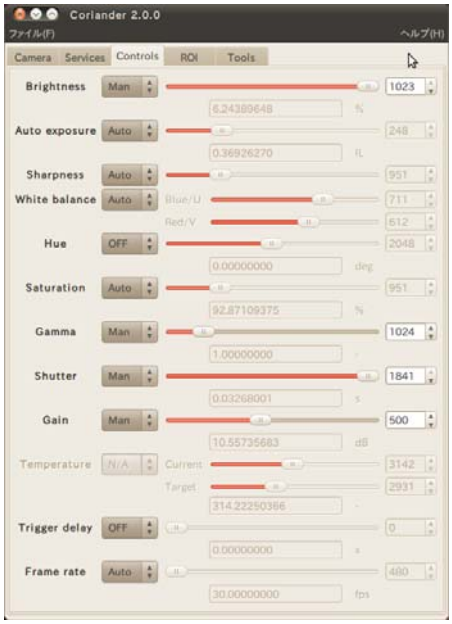


図 10　Coriander の Controls 画面

プルダウンメニューの[Man]が手動設定、[Auto]が自動設定、[OFF]が設定の無効化です。[Abs]はハードウェアのレジスタ値ではなく人間にとって分かりやすい実世界の尺度に換算した値で表示します。下表のように各カメラコントロールを設定し、ビューアの映像を確認しながら値を調節します。表の値は参考例です。

カメラコントロール	設定	値
Brightness	Man	1023
Auto exposure	Auto	---
Sharpness	Auto	---
White balance	Auto	---
Hue	OFF	---
Saturation	Auto	---
Gamma	Man	1024
Shutter	Man	1841
Gain	Man	500
Trigger delay	OFF	---
Frame rate	Auto	---

ここで得た各カメラコントロールの値を ieee1394board.0 へ記載します。このとき、Auto exposure の値は-1 としてください。ieee1394board.0 に-1 と記載すると自動で設定するという意味になります。残りのカメラも同じ値を設定します。

```
0 3
0x00b09d01009bce7b 640x480-Y(mono) 7.5fps BRIGHTNESS 1023 AUTO_EXPOSURE -1 GAMMA 1024
SHUTTER 1024 GAIN 500
0x00b09d01009bce76 640x480-Y(mono) 7.5fps BRIGHTNESS 1023 AUTO_EXPOSURE -1 GAMMA 1024
SHUTTER 1024 GAIN 500
0x00b09d01009bce7f 640x480-Y(mono) 7.5fps BRIGHTNESS 1023 AUTO_EXPOSURE -1 GAMMA 1024
SHUTTER 1024 GAIN 500
```

※ 紙面の都合で改行しています。実際はカメラ一台分の設定を一行で記述します。

以上で ieee1394board.0 ファイルにカメラコントロールが記述されました。

次にカメラの並び順を設定します（図 5)。ieee1394board.0 ファイルの 2 行目以降の各行はそれぞれカメラ 1 台に対応しています。上から順に左カメラ、右カメラ、中央カメラとなるよう行を並び替えてください。ツールによって生成されたファイルの行は順不同です。接続されているカメラの位置関係が反映されていません。

再びCoriandrでCameraタブを開いてください（図 8)。現在ビューアに表示されているのは、このCameraタブのCamera Selectで選択したカメラからの映像です。同タブ内のCamera informationにあるGUIDが、この映像を送っているカメラのIDになります。

ビューアの映像に注目しながら一台ずつカメラの前に手をかざすなどして、表示されている GUID がどのカメラのものかを調べます。全てのカメラの GUID が把握できたら、それをもとに ieee1394board.0 に記載されているカメラの順序を入れ替えます。

ieee1394board.0 にカメラコントロールの値とカメラの順番が記載し終えたら、Coriander を終了してください。以上で、ieee1394bord.0 設定ファイルの作成は終わりです。作成した ieee1394board.0 は build ディレクトリに置いてください。

3. システム構成図

作業対象認識モジュール群のシステム構成図を下記に示します。

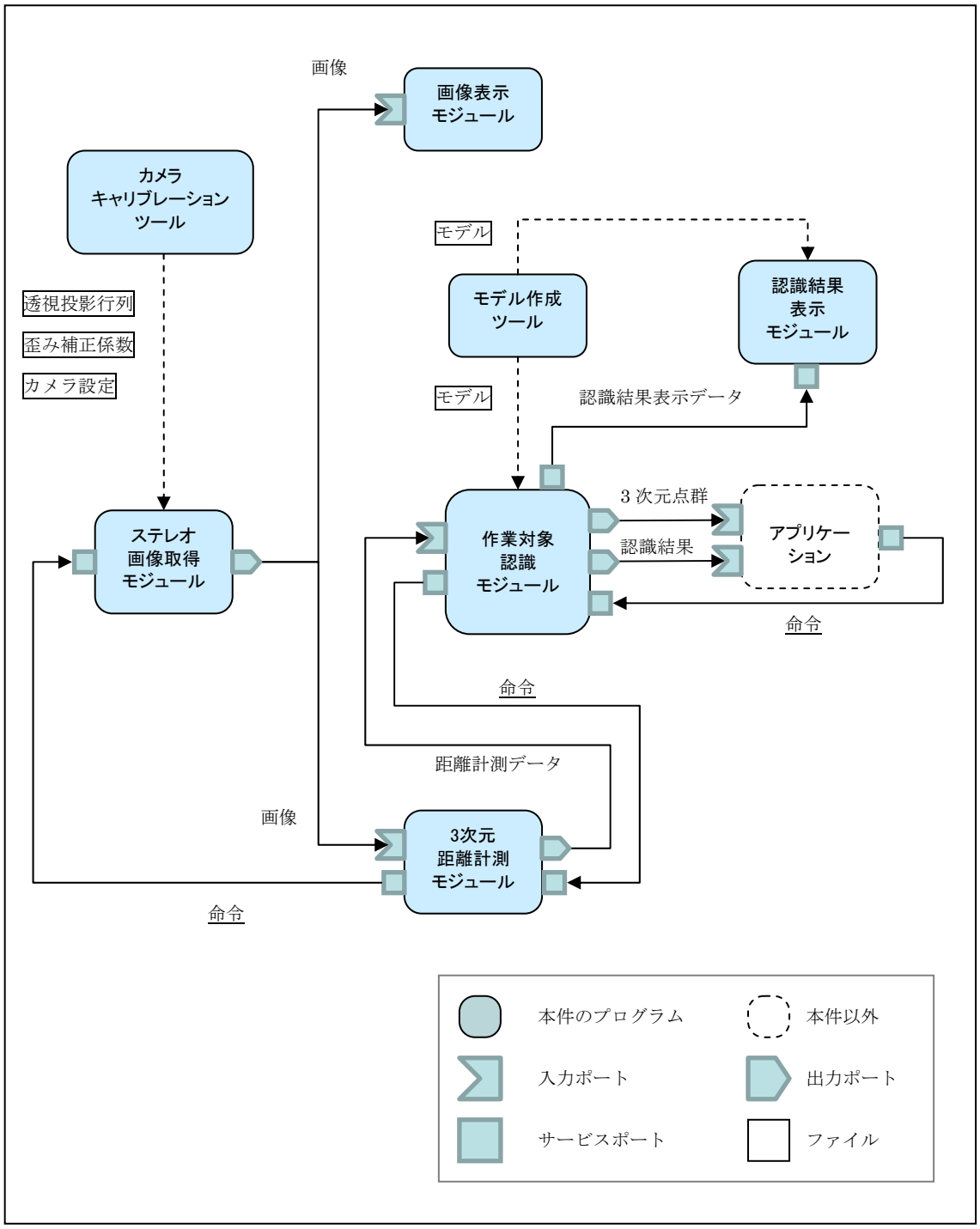


図 11　システム構成図

4. 操作方法

4.1. 作業対象認識モジュール

4.1.1. 作業対象認識モジュールの概要

作業対象認識モジュールは、モデルに基づく 3 次元の物体認識を行うプログラムです。画像中から対象物を同定し、その位置と姿勢を算出します。

作業対象認識モジュールは、次の順番で処理を行います。

1. 物体認識を実行するよう命令を受ける。
2. 距離計測データ（画像データと 3 次元点群）を送信するよう、次のモジュールに命令を送る。
3. ID 番号によって指定された対象物のモデルをファイルから読み込む。
4. 撮影されたシーンに関する距離計測データを受信する。
5. シーンから，指定されたモデルについて物体認識を実行し、位置・姿勢を同定する。
6. 認識結果と 3 次元点群をデータポートに出力する。

ユーザは作業対象認識モジュールにこれらの命令とデータが受け渡されるよう、適切な RT コンポーネントを接続する必要があります。

このモジュールに対する入力データは、作業対象認識モジュール群に含まれる次のモジュールの出力から得ることができます。

- 距離計測データ：3 次元距離計測モジュール
- モデル：モデル作成ツール

4.1.2. 作業対象認識モジュールの使い方

1. 各モジュールの起動と接続

モジュールの起動を行います。ここでは合計で四つの RT コンポーネントを起動しますが、RecognitionComp と RecognitionResultViewerComp の二つ以外は動作確認用のテスト RT コンポーネントです。

まず、コンソールを一つ開いて、三つの RT コンポーネントを起動します。

```
$ ./RecognitionResultViewerComp -f rtc.conf &
Screen = ..., ...
$ ./RecognitionComp -f rtc.conf &
$ ./SendImageComp -f rtc.conf
```

次に、もう一つコンソールを開き、SetModelIDComp を起動します。

```
$ ./SetModelIDComp -f rtc.conf
```

SendImageComp と SetModelIDComp がそれぞれキーボードから値の入力を必要とするので、このように二つのコンソールで起動を行います。他のモジュールはキーボードから値入力をする必要がないので、&を付けてバックグラウンドで走らせます。

以上で、RT コンポーネントプログラムの起動作業は終わりです。ディスプレイ上には SendImageComp の起動したコンソールと SetModelIDComp の起動したコンソールが表示されているはずです。

プログラムが正常に起動したら、eclipseを起動してRTSystemEditorの画面を開きます。起動したRTコンポーネントをリストからSystem Diagramにドラッグ&ドロップし、図 12 のように接続します。

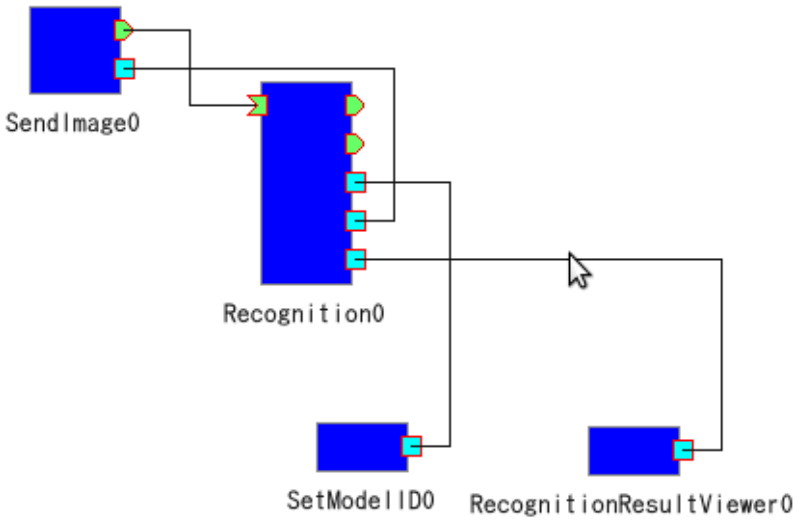


図 12　作業対象認識モジュールの接続

RecognitionComp はここで説明する作業対象認識モジュールであり、RecognitionResultViewerComp は次節で説明する認識結果を表示する RT コンポーネントです。RecogitionComp のポートの説明と各ポートに接続する RT コンポーネントについては、この節の末尾にある補足 1.をご参照ください。

RTコンポーネント間の接続ができたら、次にSystem Diagram上のRecognitionCompを選択し、Configurationタブの設定値を次のように編集します（図 13)。

- `RecogModelListPath: model/ModelList.txt`
- `RecogParameterFilePath: param/can190A.txt`

設定を編集したら適用ボタンを押してください。設定内容の意味については、この節の末尾にある補足 2.を参照してください。

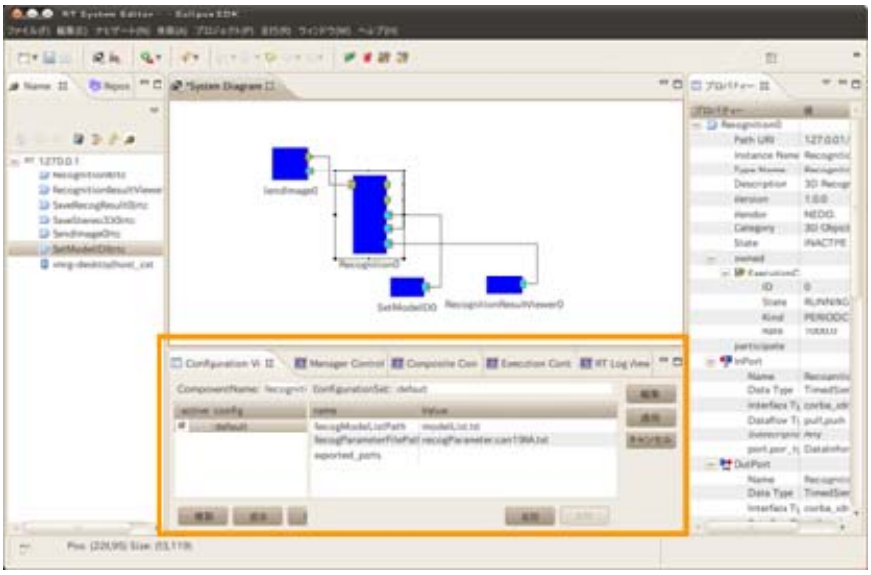


図 13　Configuration の設定

同様に RecogResultViewerComp の Configuration タブの設定値を編集します。

- `RecogModelListPath: model/ModelList.txt`

SendImageComp の設定は必要に応じて直してください。全ての RT コンポーネントを設定したら、RTSystemEditor で RT コンポーネントを Active 状態にします。

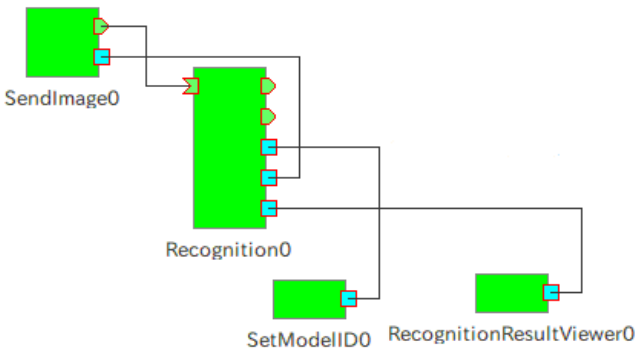


図 14　Active 状態

2.　モジュールの実行

RT コンポーネントが Active 状態になると、SetModelIDComp が標準出力に認識するモデル ID を入力するよう促すメッセージを出力します。

```
$ ./SetModelIDComp -f rtc.conf
モデル ID を入力してください：105
```

ModelList.txt に記述したモデル ID をコンソールから入力します。

入力されたモデル ID は SetModelIDComp から作業対象認識モジュール（RecognitionComp）に送られます。作業対象認識モジュールはモデル ID を受け取ると、SendImageComp に画像データの送信要求を出します。

SendImageComp が画像データ送信要求を受け取ると、認識する画像データのファイル名等を入力するよう促すメッセージを標準出力に出力します。SendImageComp は、距離計測データに相当するものを送信するテストコンポーネントなので、ダミー出力する点群の数、エラーコード、送信する画像データそれぞれのファイル名、送信するキャリブレーションデータのディレクトリ名を必要とします。コンソール画面の指示に従い、順次入力してください。

```
出力する点群数を入力してください。: 0
出力するエラーコードを入力してください。: 0
0 枚目の読み込む画像ファイル名を入力してください。: image/can190A.0.ppm
1 枚目の読み込む画像ファイル名を入力してください。: image/can190A.1.ppm
2 枚目の読み込む画像ファイル名を入力してください。: image/can190A.2.ppm
キャリブレーションデータのパスを入力してください。: calib/set0
```

エラーコードは 0（エラーなし）を入力してください。画像データは、0 枚目は左カメラ画像、1 枚目は右カメラ画像、2 枚目は中央カメラ画像になります。なお、画像データの枚数は、SendImageComp の ReadImageNum コンフィグレーションで指定した枚数になります。

認識が成功すると、作業対象認識モジュールは認識結果を認識結果出力ポートから出力します。

また、作業対象認識モジュールはサービスポートから認識結果表示コンポーネントを呼び出します。呼び出された認識結果表示コンポーネントは、ウインドウを開いて認識結果を画面に表示します。

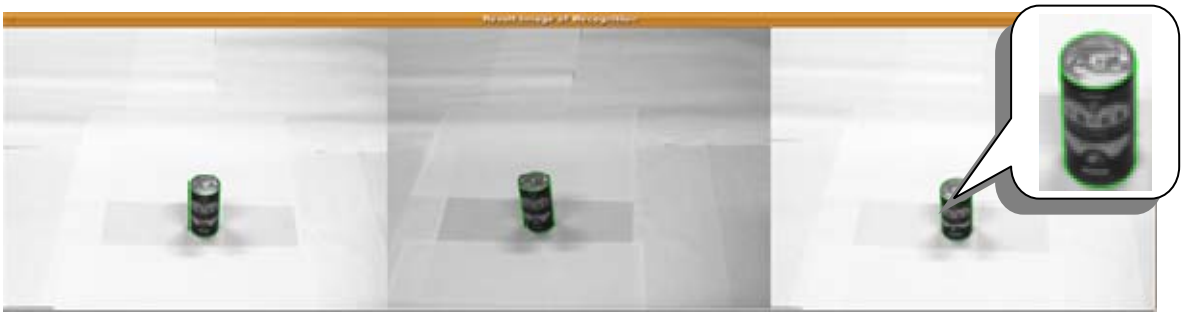


図 15　認識結果の表示

3. モジュールの終了

RT System Editor で RT コンポーネントを選択し、右クリックメニューから Exit を選択してください。

補足 1. RTC の機能：

作業対象認識モジュール群の RTC

RT コンポーネント名称	機能
RecognitionComp	作業対象物を認識する。
RecognitionResultViewerComp	認識結果を表示する。

動作確認テスト用 RTC

テスト用 RT コンポーネント名称	機能
SetModelIDComp	認識するモデルデータを設定する。コンソールからモデル ID を入力し、作業対象認識コンポーネントへ認識実行命令を送る。
SendImageComp	距離計測データ送信要求に応じて画像を送信する。作業対象認識モジュールから距離計測データの取得要求を受け取り、コンソールから点列数、画像ファイル、キャリブレーションデータを読み込んで Stereo3D 構造体形式で、作業対象認識モジュールのデータポートに送信する。

補足 2. コンフィグレーション設定：

必要に応じて RecognitionComp と RecogResultViewerComp、および SendImageComp のコンフィギュレーション設定を編集してください。各コンポーネントの編集が必要なコンフィギュレーション設定を下記に示します。なお、コンポーネントを一旦 Active 状態にした後にこれらのコンフィギュレーションを変更する場合、コンポーネントを一度 Deactivate してから行い、再度 Activate してください。

RecognitionComp のコンフィグレーション設定項目	
RecogModelListPath	認識に使用するモデルと、モデル ID の組を一覧にしたテキストファイルのファイル名を指定します。
RecogParameterFilePath	認識に使用する各種パラメータを記述したテキストファイルのファイル名を指定します。

RecogResultViewerComp のコンフィグレーション設定項目	
RecognitionResultViewerComp	認識に使用するモデルと、モデル ID の組を一覧にしたテキストファイルのファイル名を指定します。

SendImageComp のコンフィグレーション設定項目	
ReadImageNum Configuration	送信する画像枚数を指定します。 2 眼では 2 枚を、3 眼では 3 枚を指定してください。

コンポーネントごとに、全ての設定が終わったら適用ボタンを押して設定を適用します。

4.2. モデル作成ツール

4.2.1. モデル作成ツールの概要

モデル作成ツールは、作業対象認識モジュールで利用するモデルを作成するためのコマンドラインプログラムです。このモデルは、対象物に関する形状を記述したデータです。なお、現在のバージョンのモデル作成ツールは、円柱と直方体のみをサポートしています。

4.2.2. モデル作成ツールの使い方

1. モデル作成ツールの使用方法

モデル生成を行うには、設定ファイルに生成するモデルの設定を記述し、下記のように実行します。

```
$ ./VGRModeler –i in.txt –o out.wrl
```

ここで in.txt はテキスト形式のモデル生成用設定ファイル、out.wrl は生成されたモデルが出力されるファイルです。任意のファイル名を付けてください。

2. 直方体モデルの生成方法

直方体のモデル作成は以下の手順で行います。

テキストエディタを立ち上げ、モデル生成用設定ファイルを作成します。ここでは下記の内容でテキストファイルを作成し、ファイル名を box_rtvcm.txt とします。「#」で始まる行はコメントとして無視されますので、入力する必要はありません。

```
#直方体
#50×30×100 mm

#幅(x):50mm、奥行き(y):30mm、高さ(z):100mm の直方体（重心位置:0,0,0）の作成
B 50,30,100
```

作成したモデル生成用設定ファイルを使ってモデルを生成します。

```
$ ./VGRModeler -i box_rtvcm.txt -o box_rtvcm.wrl
The box created.
$
```

モデルが生成されると、その旨のメッセージが表示されます。

3. 円柱モデルの生成方法

円柱のモデル作成を同様に行います。ここでは下記内容のテキストファイルを作成し、ファイル名を cylinder_rtvcm.txt とします。

```
#円柱
#高さ 100 mm, 直径 50 mm

#半径:25mm、高さ:100mm の円筒（重心位置:0,0,0）の作成
C 25,100
```

作成したモデル生成用設定ファイルを使ってモデルを生成します。

```
$ ./VGRModeler -i cylinder_rtvcm.txt -o cylinder_rtvcm.wrl
The cylinder created.
$
```

モデルが生成されると、その旨のメッセージが表示されます。

4. モデルの確認方法

生成されたモデルを画像で確認します（図 16、図 17）。

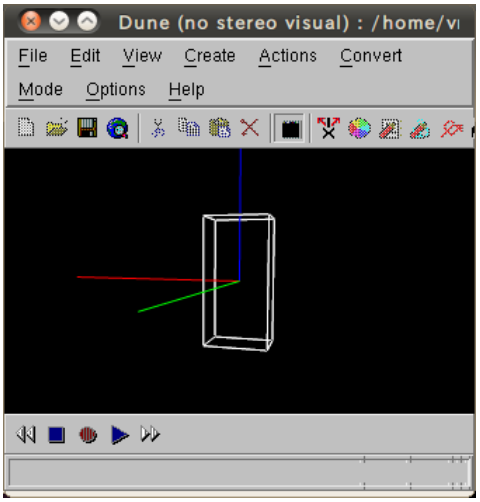


図 16　直方体の例

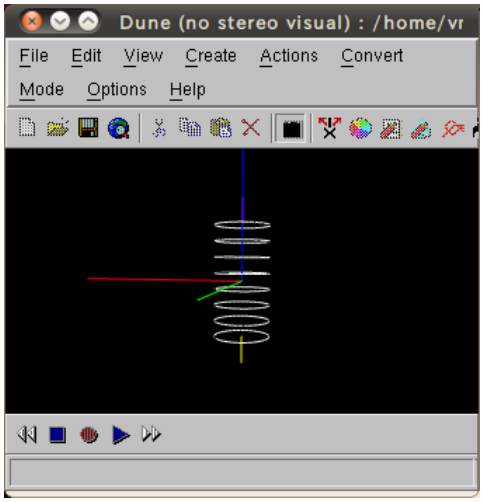


図 17　円柱の例

ここでは説明のために Synaptic でインストールできる VRML エディタの WhiteDune を使用しています。なお、WhiteDune の場合、初期状態ではモデルのフレームが背景色と同じ黒色で表示されます。その際は表示する VRML ファイルを開いた後、WhiteDune の中央付近にあるツールバーの Background ボタンを押してください。フレームが白色で表示されます。

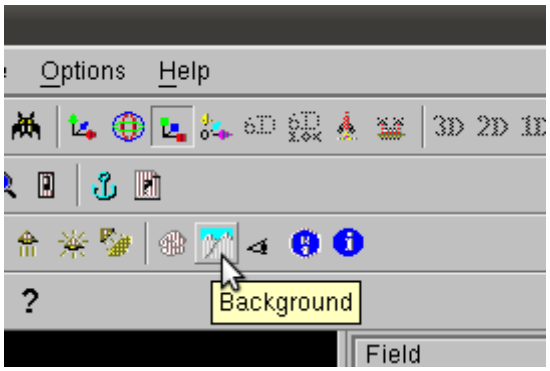


図 18　WhiteDune の Background ボタン

4.2.3. 用途に応じてモデルの原点と軸方向を変える

モデルに回転情報を付与する場合（図 19）は、モデル生成用の設定ファイルに下記のように記述します。

```
#直方体
#20×10×20 mm

#幅(x):20mm、奥行き(y):10mm、高さ(z):20mm の直方体（重心位置:0,0,0）の作成
B 20,10,20

#x 軸方向に 90 度、y 軸方向に 45 度、z 軸方向に 45 度回転
R 90,0,0
R 0,45,0
R 0,0,45
```

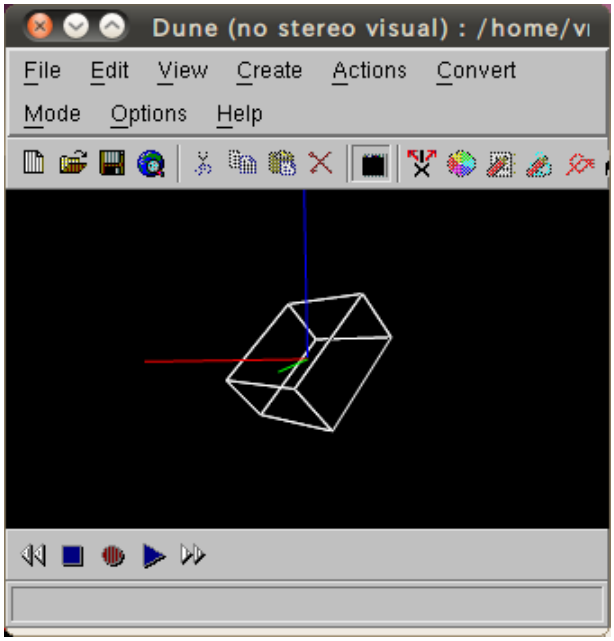


図 19　回転の例

モデルに移動情報を付与する場合は、モデル生成用の設定ファイルに下記のように記述します。

```
#直方体
#20×10×20 mm

#幅(x):20mm、奥行き(y):10mm、高さ(z):20mm の直方体（重心位置:0,0,0）の作成
B 20,10,20
#重心位置の移動量(x,y,z)
T 0,0,50
```

4.3. 3 次元距離計測モジュール

4.3.1. 3 次元距離計測モジュールの概要

3 次元距離計測モジュールは、物体の表面を点の集まりとして計測する RT コンポーネントです。別途、撮影されたステレオ画像から、画像処理によって各点の 3 次元座標を算出します。こうして求められた 3 次元点群は、距離計測データに含まれる 3 次元点群として出力されます。

4.3.2. 3 次元距離計測モジュールの使い方

1. モジュールの使用方法

3 次元距離計測モジュールは、モジュールが置かれているディレクトリに設定ファイル“measure3d_config.d”を必要とします。以下にそのサンプルがあります。

~/opt/OpenVGR/src/module/component/Measure3D/measure3d_config.d

これをモジュールが置かれているディレクトリにコピーし、適宜、値を編集してください。このとき、この設定ファイルの値は省略できませんので、全てのパラメータを指定してください。

設定ファイルの例を以下に示します。

```
preFilterType 0
preFilterSize 33
preFilterCap 33
SADWindowSize 33
minDisparity 0
numberOfDisparities 128 // must be divisible by 16 !
textureThreshold 10
uniquenessRatio 15
speckleWindowSize 0
speckleRange 10
trySmallerWindows 1
```

このように、各行に″パラメータ名〈空白〉値〈空白〉[コメント]″を書きます。記述のきまりとして、〈空白〉は 1 個以上の半角スペースまたはタブになります。値は全て整数です。数値の次の空白以降は改行まで無視されますので、自由な記述が行えます。

設定ファイル“measure3d_config.d”に記述するパラメータでウィンドウサイズ、視差

に関するものは下記になります。

パラメータ名	説明
SADWindowSize	ステレオ相関法のウィンドウサイズ。5 以上の値。
minDisparity	最小視差。
numberOfDisparities	視差の探索範囲。 最大視差 － 最小視差 ＋ 1 16 の倍数でないとエラーになる。

3 次元距離計測モジュールは、コンソールから下記のように起動します。このモジュールは他の章で解説する RT コンポーネントと接続して利用します。ここでその説明は割愛します。他の RT コンポーネントとの接続はシステム構成図を参照してください。

```
$ ./Measure3DComp –f rtc.conf
```

4.4. ステレオ画像取得モジュール

4.4.1. ステレオ画像取得モジュールの概要

ステレオ画像取得モジュールは、複数のカメラを用いてステレオ画像を撮影する RT コンポーネントです。

4.4.2. ステレオ画像取得モジュールの使い方

1. 準備

カメラの設定値を記録されているファイル ieee1394board.0 と、カメラの幾何学的な関係が記録されているキャリブレーションデータを用意する必要があります。最初に p.12 の手順に従って ieee1394board.0 を作成してください。この設定ファイルは build ディレクトリに置いてください。

2. モジュールの起動と接続

画像取得モジュールはMultiCameraCompという名前のRTコンポーネントです。ここでは MultiCameraComp、MultiCameraTriggerCompという二つのRTコンポーネントを起動して互いに接続します。MultiCameraTriggerCompは画像取得モジュールを動作させるサンプルのRTコンポーネントです。説明のためここではMultiCameraTriggerCompが、図11の 3 次元距離計測モジュールに代わって命令を送ります。

MultiCameraComp を起動します。

```
$ ./MultiCameraComp -f rtc.conf
```

同様に MultiCameraTriggerComp を起動します。

```
$ ./MultiCameraTriggerComp -f rtc.conf
```

各 RT コンポーネントが正常に起動したら、RTSystemEditor で System Diagram を開き、下図のようにポートを接続します。

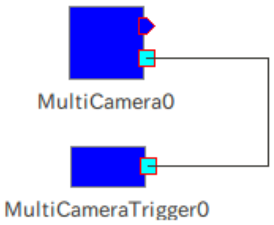


図 20　ステレオ画像取得モジュールの接続

3.　モジュールの実行

接続ができたら、RTSystemEditorで、All Activateしてください。両方のコンポーネントをActive状態にすれば、画像が取得されます。ここでは接続と動作のみ確認します。なお、取得した画像を確認するには、画像表示モジュールが必要です。画像取得モジュールへ画像表示モジュールを接続する方法は、このまま「4.5画像表示モジュール」を読み進めてください。

4.　モジュールの終了

全てのコンポーネントを Deactivate し、キーボードから Ctrl-C を入力してそれぞれ終了させてください。

4.5. 画像表示モジュール

4.5.1. 画像表示モジュールの概要

画像表示モジュールは画像を表示するプログラムです。ステレオ画像取得モジュールが出力するステレオ画像をビューアに表示します。

4.5.2. 画像表示モジュールの使い方

1. 各モジュールの起動と接続

MultiDispComp を起動して、ステレオ画像取得モジュールに接続します。ステレオ画像取得モジュール（MultiCameraComp、MultiCameraTriggerComp）は起動および接続が済んでいるものとします。

MultiDispComp を起動します。

```
$ ./MultiDispComp -f rtc.conf
```

プログラムが正常に起動したら、RTSystemEditor で System Diagram を開き、下図のようにポートを接続します。

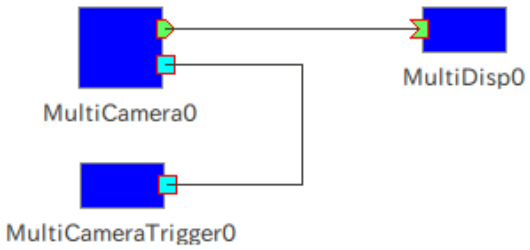


図 21 画像表示モジュールの接続

2. モジュールの実行

接続ができたら、RTSystemEditor で、All Activate してください。3 つのコンポーネントを全て Active 状態にすれば、画像表示モジュールがステレオ画像取得モジュールによって撮影された画像を表示します。

3. モジュールの終了

全てのコンポーネントを Deactivate し、キーボードから Ctrl-C を入力してそれぞれ終了させてください。

4.6. カメラキャリブレーションツール

4.6.1. カメラキャリブレーションツールの概要

カメラキャリブレーションツールは計測に必要なカメラのキャリブレーションデータを作成するプログラムです。複数のカメラ画像からチェスボードパターンを検出し、そこで得た格子点座標情報をもとに、各カメラの内部パラメータとカメラ間の相対位置姿勢を算出します。

カメラキャリブレーションツールは二つのプログラムから成ります。ツール構成を下記に示します。

名称	機能
カメラキャプチャ・チェスボードパターン検出 GUI プログラム	・ 複数のカメラ画像のライブ表示をします。 ・ チェスボードパターンを撮影し、格子点座標を検出します。
マルチカメラキャリブレーションプログラム	・ チェスボードパターンの検出結果を用いて、カメラのキャリブレーションデータを計算します。

■カメラキャプチャ・チェスボードパターン検出 GUI プログラムの詳細
設定ファイル：

・ ieee1394board 形式設定ファイル

（ファイルの詳細については「2.3.3必要に応じて設定する項目」の「■ieee1394board.0について」をご参照ください。）

出力ファイル形式（検出格子点座標列）：

```
MLTCLB  <バージョン番号>
<パターン間隔[mm]>  <パターン回転周期[deg]>
<観測回数>  <カメラ台数>
<1 カメラ，１回目検出点数>
<参照平面座標 x[mm]>  <参照平面座標 y[mm]>  <画像座標 x[pixel]>  <画像座標 y[pixel]>
：
<2 カメラ，１回目検出点数>
：
<1 カメラ，2 回目検出点数>
```

■マルチカメラキャリブレーションプログラムの詳細

入力データ：

・カメラキャプチャ・チェスボードパターン検出 GUI より出力された検出格子点データ

出力：

・各カメラの内部パラメータ（3×3 行列)、外部パラメータ（3×4 行列）

・各カメラの歪み補正係数

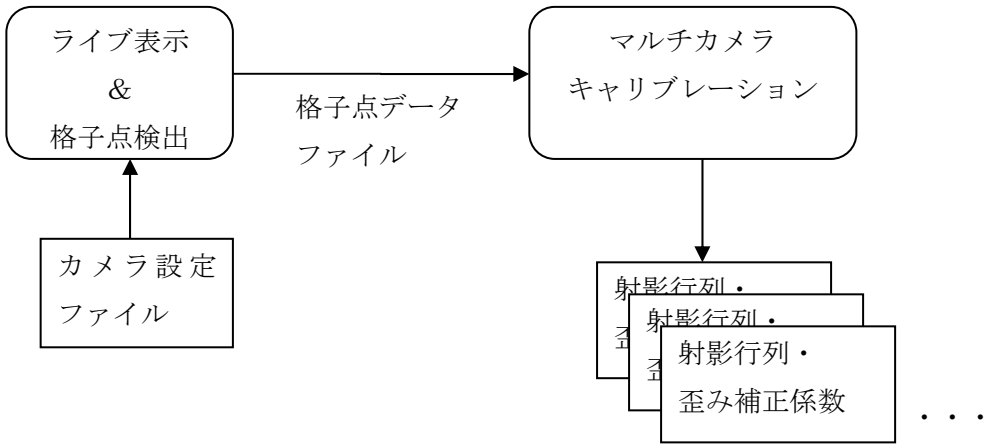


図 22　カメラキャリブレーションツールの概要

4.6.2. カメラキャリブレーションツールの使い方

1.　キャリブレーション板の準備と使用方法

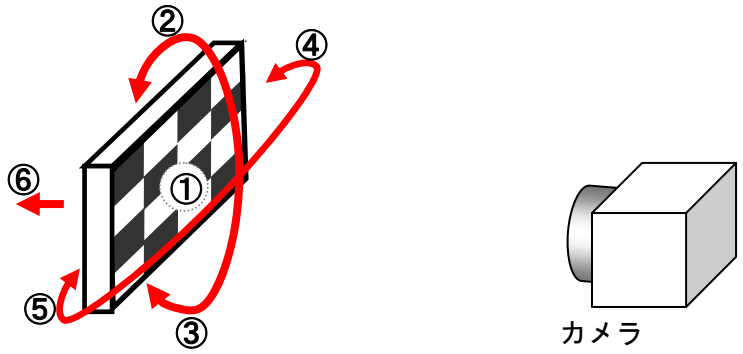
配布物中にあるキャリブレーション用チェスボードパターンの画像を印刷して可能な限り平らな板に貼り、キャリブレーション板を作ります。

パターンの画像が、下記 URL から入手できます。

http://opencv.jp/sample/pics/chesspattern_7x10.pdf

これを印刷し、板に貼ると、7 行 10 列のキャリブレーション板として使えます。マス目のサイズは定規などで実測してください。

キャリブレーションを行うにあたりキャリブレーション板を図 23のように動かして、検出プログラムで撮影を繰り返します。ここでは 6 枚の画像を撮影します。



番号	動作
①	カメラとキャリブレーション板を並行にする。
②	①の状態から上に向かって傾ける。
③	①の状態から下に向かって傾ける。
④	①の状態から左に向かって傾ける。
⑤	①の状態から右に向かって傾ける。
⑥	①の状態から後方へ移動する。

図 23　カメラのキャリブレーション

なお撮影にあたって、チェスボードパターン以外の、検出の妨げとなる余分なものが画像に写らないようにしてください。

2.　カメラキャプチャ・チェスボードパターン検出 GUI プログラムの使用方法

カメラキャプチャ・チェスボードパターン検出 GUI プログラムは ichimatsu という名称です。

① プログラム ichimatsu を起動します。

```
$ ./ichimatsu -s 24 –g 7x10 -o calib.xcdata
```

プログラム ichimatsu に指定できるコマンドラインオプションは下記の通りです。

パラメータ	役割
-o <filename>	検出結果を保存するファイル名を指定。
-s <size>	チェスボードパターンの一つのマス目の一辺の長さを指定。 単位は mm。デフォルト値は 20。

-g <row>x<col>	チェスボードパターンの交点の数を行と列で指定。デフォルト値は 8x8
-d <dirname>	検出に使用した画像を指定ディレクトリに保存。デフォルト値は指定なし（保存しない）。

プログラムが正常に起動すると、カメラの台数に応じた数のビューアが開きます。プログラムは ieee1394board.0 に記載したカメラの順にビューアを割り当てますので、それぞれのビューアのタイトルバーの表示"Camera 0"、"Camera 1"、"Camera 2"は、左カメラ画像、右カメラ画像、中央カメラ画像に対応しているはずです。一度各カメラに手をかざしてみるなどして、ビューアの表示とカメラの位置に誤りがないか簡単なチェックを行ってください。ieee1394board.0 はカレントディレクトリにないといけません。

ビューアの表示が大きすぎて全部のカメラ画像を同時に見るのが困難なときは、「2」～「4」の数字キーを押すと数字に応じて縮小表示します。さらに「－」、「＋」キーで大きさの微調整ができます。

キャリブレーション板がカメラレンズに対してなるべく正面になるようチェスボードパターンの面をカメラに向けて、一枚目の正面の撮影を行います（図 23）。

② キーボードで「c」キーを押し、パターンの検出を開始します。プログラムの検出結果がビューアの画像に重ねて描画されます。

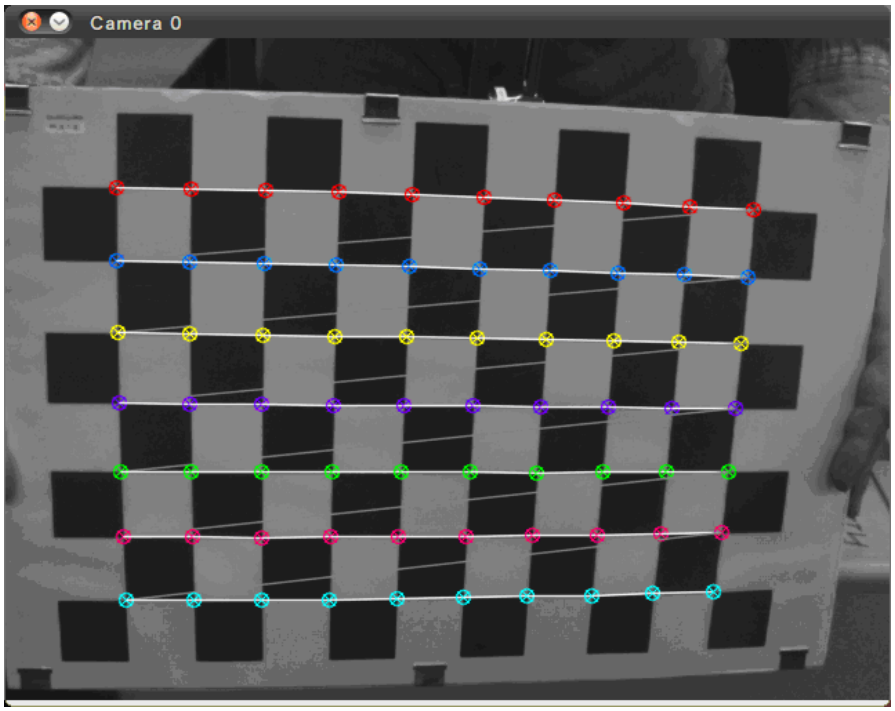


図 24　検出されたチェスボードパターン 7×10 の正面画像

③ どのカメラ映像でもチェスパターンの全ての角に丸印が付され、かつビューア内に最大の状態でチェスボードパターンが映っていることが確認できたら、「リターン」キーを押して撮影を行います。「リターン」キーは、３つのカメラビューアのいずれかが選択された(フォーカスが当たっている)状態で押して下さい。
撮影できると、コンソールに「added the detected points. 」のメッセージが出力されます。
④ 正面が撮影できたら、次にキャリブレーション板を上に傾けた状態を作ります。このとき余り傾け過ぎると検出ができず丸印が消えるので、注意してください。手順③に従い撮影を行います。さらにキャリブレーション板の向きを変えて撮影することを繰り返し、正面、上、下、左、右、後方の合計 6 回観測させます（図 23)。

ここで大切なことはボードのカメラに対する向きが 3 種類以上あることです。ただし、互いに平行になるものは 1 種類と数えます。結果をより確実にするため、撮影回数を増やしてもかまいません。

⑤ 十分な検出データを得たら保存を行います。「Shift」キーと「w」キーを同時に押すと出力ファイルに検出格子点データが書き込まれます。

⑥ 「q」キーを押して終了します。

以上で、カメラキャプチャ・チェスボードパターン検出 GUI プログラムでの作業は終了です。

3. マルチカメラキャリブレーションの使用方法

マルチカメラキャリブレーションは multicalib という名称です。

プログラム multicalib を実行します。

```
$ ./multicalib -o camera_calib.yaml calib.xcdata
```

これによってカメラのキャリブレーションデータが得られます。出力ファイルは拡張子.yaml で-o オプションを付けて指定します。計算が正しく終了した目安は、最後に表示される再投影誤差(error)の値が 0.2 程度になることです。この誤差が大きな場合には、キャリブレーション板の位置や姿勢などを変えることで、誤差が小さくなる可能性があります。

もし、キャリブレーションがうまくできない場合は以下の点に注意して下さい。

- 平面の見せ方
  - なるべく画面いっぱいにキャリブレーション板が写るようにする。
  - 3 次元認識を行う範囲よりも大きく動かす。
  - 画像の中心付近だけでなく、まんべんなく撮影されるようにする。
  - 色々な方向にしっかりと傾ける。
  - 検出された点がずれていないか確認する。
  - 観測時に画像がブレないよう、キャリブレーション板を静止させる。
- 平面の精度
  - 平面板に歪みがないか確認する。
  - パターンが正方形(4 辺の長さ、角度)を確認する。

以上でカメラキャリブレーションの作業は終了です。プログラム multicalib によって生成されたキャリブレーションデータのファイルを配布物の仮のキャリブレーションデータファイルと置き換えて使用してください。

4.7. rtshell を利用した実行スクリプト例について

毎回 RTSystemEditor を使って RT コンポーネントの起動を行うのは大変です。ひとつの方法として、rtshell を使ったシェルスクリプトを用意すると RT コンポーネント（以下コンポーネント）の実行が簡単になります。OpenVGR のセットはこのスクリプトの例を含んでいます。以下でスクリプト例の操作手順について説明します。ここでは rtshell-3.0 の使用を前提とします。

4.7.1. 準備

OpenVGR モジュール群のビルドと rtshell のインストールをします。OpenVGRモジュール群のビルドについては本書の「2.2 インストール」から「2.3.1 サンプルのコピー」を参照し、buildディレクトリでの make まで実行して下さい（既に終わっていれば不要)。rtshell のインストールについて次に記します。

rtshell のインストール手順

■ OpenRTM-aist-1.0.0 Python 版をインストールする
http://www.openrtm.org/OpenRTM-aist/download/install_scripts/pkg_install_python_ubuntu.sh
より pkg_install_python_ubuntu.sh をダウンロードし実行

■ rtshell, rtctree, rtsprofile をインストールする

以下のソースコードをダウンロードし展開する

http://www.openrtm.org/pub/OpenRTM-aist/tools/1.0.0/rtshell-3.0.0.tar.gz

http://www.openrtm.org/pub/OpenRTM-aist/tools/1.0.0/rtctree-3.0.0.tar.gz

http://www.openrtm.org/pub/OpenRTM-aist/tools/1.0.0/rtsprofile-2.0.0.tar.gz

展開したら以下のように各ソースディレクトリのインストールスクリプトを実行する

```
$ cd rtsprofile-2.0.0
$ sudo python setup.py install
$ cd ..
$ cd rtctree-3.0.0
$ sudo python setup.py install
$ cd ..
$ cd rtshell-3.0.0
$ sudo python setup.py install
Generate documentation? n
```

rtshell-3.0.0 のインストールスクリプトでは Generate documentation? と入力待ちとなるので、n と入力する

■ ログインディレクトリの .bashrc に以下を追加する

```
export RTCTREE_NAMESERVERS="localhost"
source /usr/local/share/rtshell/shell_support
```

4.7.2. 実行方法

ターミナルを開き、OpenVGR/example/script ディレクトリに移動します。ここに静止画像での物体認識をするスクリプト stilltest.sh と、接続された 1394 カメラでステレオ画像を撮影して物体認識をするスクリプト captrecog.sh があります。 captrecog.sh はお使いの環境でのカメラ設定ファイルの作成やキャリブレーション、物体モデルが必要なため、すぐに正しい認識結果を得ることはできません。最初は stilltest.sh で付属静止画像の認識を試してください。

4.7.2.1. 付属静止画像の認識

script ディレクトリで以下のようにタイプします。

```
$ ./stilltest.sh
```

すると Recognition Progress というタイトルのターミナルが起動し、図 25のようなメッセージが表示されます。

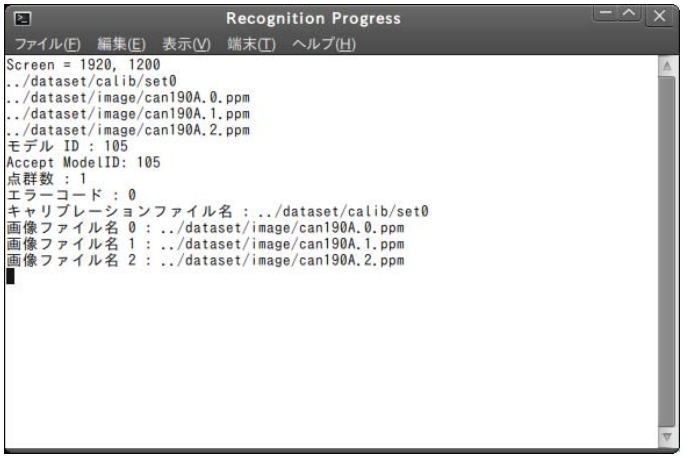


図 25　認識経過表示ターミナル

少し遅れて Result Image of Recognition というタイトルのウインドウが現れ、物体認識の結果を表示します。図 26のように、物体が撮影された画像にモデルの輪郭線が投影され、それが物体と一致していれば正しく認識されたことになります。

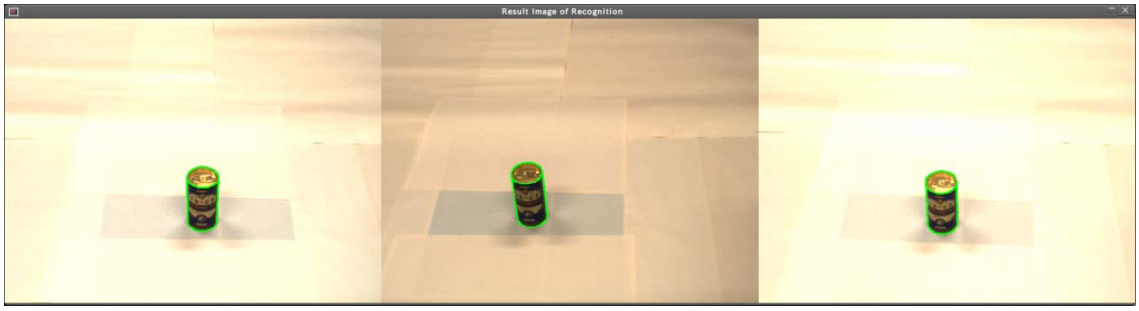


図 26　認識結果表示ウインドウ

このとき、stilltest.sh を起動したターミナルには

```
hit enter-key after checking the result.
```

というメッセージが表示されているはずです。このターミナルを選択し Enter キーを押すと次のステレオ画像セットの認識を実行します。 script ディレクトリに testlist.txt というファイルがあります。本例では Enter キーを押していきこのファイルに記された 7 個の画像セットを認識するとスクリプトが終了します。つまり stilltest.sh は testlist.txt の各行を順に取り出し、指定されたデータを読んで物体認識を実行します。この各行には以下の項目がスペースで区切られて記されています。

・認識パラメータファイル名
・モデル ID
・キャリブレーションデータディレクトリ
・キャリブレーションデータファイル名
・入力画像ディレクトリ
・入力画像０ファイル名（左カメラ画像）
・入力画像１ファイル名（右カメラ画像）
・入力画像２ファイル名（中央カメラ画像）
・エラーコード（０にしておく）
・点群数（１にしておく）

モデル ID は OpenVGR/example/model/ModelList.txt の各行先頭に記述された番号を使用します。また、画像などのデータは主に OpenVGR/example/dataset 下に置かれています。

上記のパラメータは script/substill にある comact.sh スクリプトへの引数として渡されます。なお、script/substill ディレクトリには stilltest.sh から呼び出されるサブスクリプトが置いてあり、それぞれ以下のような役割になっています。

- **comup.sh　　コンポーネント起動**
- **comcon.sh　コンポーネント接続**
- **comact.sh　コンポーネント活性化**
- **comdeact.sh コンポーネント不活性化**
- **comdown.sh　コンポーネント終了**

stilltest.shからコンポーネント起動～活性化までのサブスクリプトを実行したときは RTSystemEditor の System Diagramで見ると図 27のようになります。

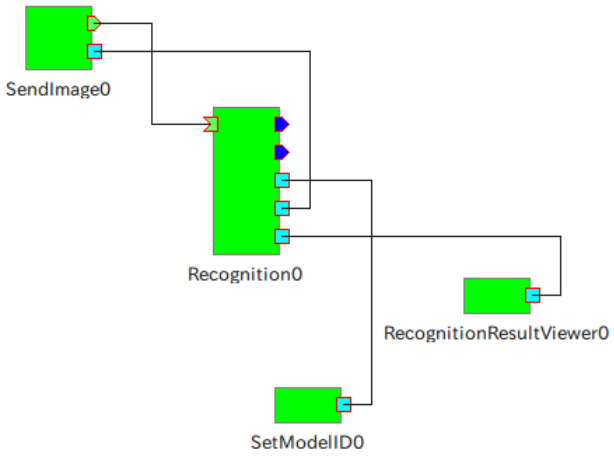


図 27　stilltest.sh の実行時コンポーネント状態

stilltest.sh を途中で終了するときは stilltest.sh の実行ターミナルで Ctrl-C を押し、その後以下のように二つのスクリプトを順に実行します。

```
$ ./substill/comdeact.sh
$ ./substill/comdown.sh
```

4.7.2.2. カメラキャプチャ画像の認識

先に述べたように、captrecog.sh を使ってカメラキャプチャから物体認識を行うことができます。実際に動作させるにはカメラの設定ファイルとキャリブレーションデータ、認識物体のモデルを作成しておくことが必要です。カメラの設定は本書の「2.3.3 必要に応じて設定する項目」、キャリブレーションは「4.6 カメラキャリブレーションツール」、モデル作成は「4.2 モデル作成ツール」をそれぞれ参照して行ってください。作成したカメラ設定ファイル ieee1394board.0 とキャリブレーションデータ camera_calib.yaml は build ディレクトリに置きます。

script ディレクトリで captrecog.sh を実行すると、ターミナルには図 28のようなメッセージが表示されます。

図 28　captrecog.sh の実行開始時メッセージ

また、 SetModelID というタイトルのターミナルが起動します。
このとき RTSystemEditor の System Diagram で見ると図 29のようになります。

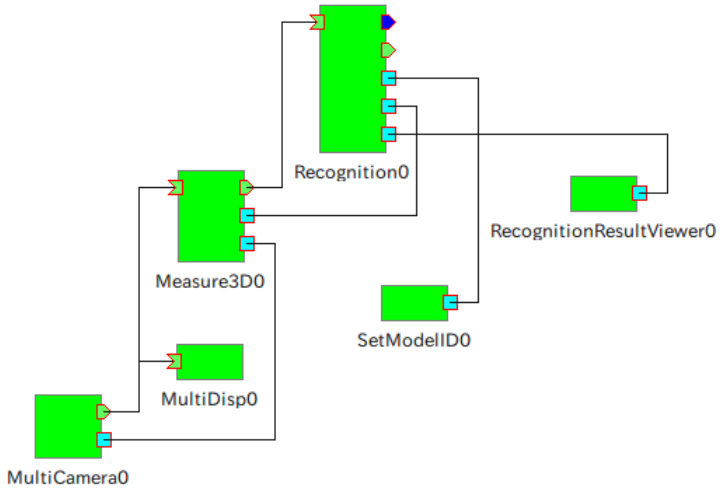


図 29　captrecog.sh の実行時コンポーネント状態

SetModelID ターミナルには以下のプロンプトが表示されます。

```
モデル ID を入力してください。 :
```

ここにモデル ID 番号を入力すると、カメラ画像取得と物体認識が行われます。モデル ID 番号は静止画像の認識で用いたものと同じです。自作のモデルを使うにはモデルのファイルを example/model ディレクトリに追加し、example/model/ModelList.txt に ID 番号とそのモデルファイル名（パス）を登録します。詳しくは機能仕様書の RecognitionComp（作業対象認識 RTC）設定ファイルの項を参照下さい。

モデル ID 番号を入力後、キャプチャした画像が image0, image1, image2 というタイトルのウインドウにそれぞれ表示されます（図 30，ステレオカメラが 3 台構成の場合）。以下のスクリーンショットでは各キャプチャ画像が整列していますが、実際には位置が不定でウインドウが重なった表示となります。

図 30　captrecog.sh 実行時の image0, image1, image2 ウインドウ

認識結果は Result Image of Recognition というタイトルのウインドウ（図 31）に表示されます。

図 31　captrecog.sh 実行時の認識結果ウインドウ

SetModelID ターミナルではプロンプトが再び表示されるので、モデル ID をまた入力して別の認識を続けることができます。実行を終了するときは、captrecog.sh の実行ターミナルで Enter キーを押します。

ところで、カメラ設定やキャリブレーションをして captrecog.sh を実行しても認識結果が正しくなかったり、全く出ないかもしれません。キャリブレーションなどが正しい場合は、スクリプトが参照している認識パラメータファイル example/dataset/param/w02.txt を修正して改善を試みます。認識パラメータの詳細は機能仕様書の RecognitionComp（作業対象認識 RTC）設定ファイルの項を参照下さい。

captrecog.sh は以下のように debug というオプションを与えて実行すると中間結果の画像を表示します。

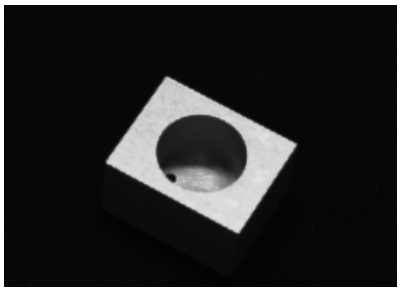
```
$ ./captrecog.sh debug
```

これらの画像を参考にして、認識パラメータの調節を行うことができます。中間結果表示中はキー入力を待っており実行が一時停止しているので、継続するには画像表示ウインドウを選択し、Enter キーを押してください。すると次の中間結果画像に切り替わります。ただし、最後の中間結果画像については Enter キーを押しても表示が消えません。次の認識を実行するか、スクリプトを終了したときに消えます。これらの表示した画像や３次元復元点は、script ディレクトリに以下の表の名前でファイルとして保存されます。拡張子の前にある番号[012]は、0～2 のカメラ ID またはステレオ組み合わせの識別番号です。なお、モデルが直方体の場合には vertex3D[012].txt が、モデルが円柱の場合には circles3D[012].ppm と circle3D[012].txt が生成されます。

ファイル名	内容
src[012].pgm	認識に使用するグレイスケール化画像
edge[012].pgm	エッジ抽出画像
line[012].ppm	直線検出結果画像
Line2Vertex[012].ppm	画像中の頂点検出画像
ellipse[012].ppm	画像中の楕円検出画像
circles3D[012].ppm	３次元復元された円検出結果画像
vertex3D[012].txt	３次元復元された頂点，端点１，端点２の座標
circle3D[012].txt	３次元復元された円の中心と周の点列座標

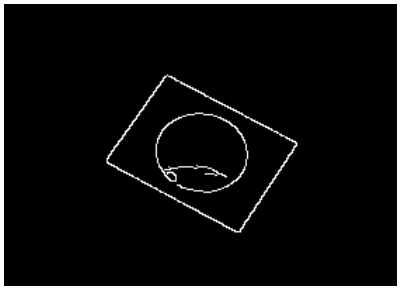
以降で中間結果画像の主なファイルについて、例を示して説明します。(画像は左カメラ・第１ステレオ組み合わせのみ)

▼ src[012].pgm



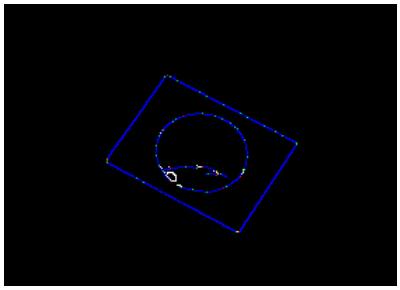
入力された画像の確認に利用します。

▼ edge[012].pgm



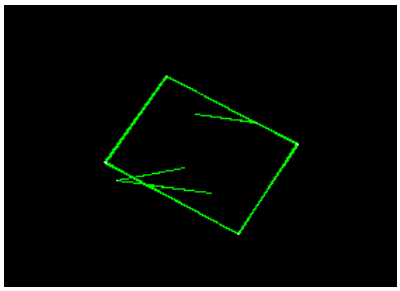
認識パラメータファイルのエッジ抽出パラメータ EdgeStrength の調節に利用します。

▼ line[012].ppm



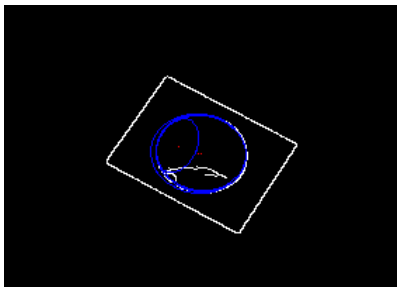
もとの輪郭線は白で、画像から検出した直線は青で描画されます。直線の始点は緑、終点は赤です。認識パラメータの直線あてはめ誤差 MaxErrorOfLineFit と区間重複比率 OverlapRatioLine の調節に利用します。

▼ Line2Vertex[012].ppm



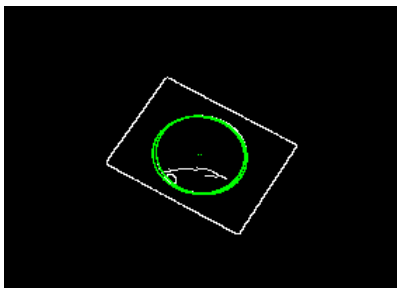
頂点を構成する直線は緑、頂点は白で描画されます。
頂点検出のためのパラメータ MinLengthLine2D と HDMax の調節に使用します。

▼ ellipse[012].ppm



もとの輪郭線は白で、画像から検出した楕円は青で描画されます。楕円の中心は赤です。
得られた２次元楕円情報の確認に利用します。

▼ circles3D[012].ppm



もとの輪郭線を白で、３次元復元できた円とその中心を緑で描画します。得られた 3 次元円情報の確認に利用します。

4.7.3. スクリプトが使用するコンフィグレーションパラメータ

スクリプト例で使用するテスト用コンポーネントの SetModelIDComp と SendImageComp は、デフォルト設定で実行するとターミナルから対話的に画像ファイル名などの情報を取得します。しかし、コンフィグレーションパラメータを使うとこれらのパラメータに対し非対話的に情報を与えることができます。また、認識および結果表示のコンポーネントである RecognitionComp と RecognitionResultViewerComp について認識パラメータファイル名，モデルリストファイル名，デバッグ情報の出力有無を、カメラキャプチャと画像表示の MultiCameraComp, MultiDispComp にはキャリブレーションファイル名，カメラ設定ファイル名，画像保存の有無を指定できます。

スクリプトでは rtshell の rtconf コマンドでコンフィグレーションパラメータの設定を行います。上記のコンポーネントについて、設定するコンフィグレーションパラメータを下に示します。

SetModelIDComp のスクリプト設定項目（stilltest.sh のみ）	
ModelID	モデル ID 番号

SendImageComp のスクリプト設定項目（stilltest.sh のみ）	
CalibDir	キャリブレーションデータディレクトリ
CalibFile	キャリブレーションデータセット名／ファイル名
ImageDir	画像データディレクトリ
ImageFile0	左カメラ画像ファイル名
ImageFile1	右カメラ画像ファイル名
ImageFile2	中央カメラ画像ファイル名
OutputPointNum	出力点群数（１を設定）
ErrorCode	エラーコード（０を設定）

RecognitionComp のスクリプト設定項目	
RecogParameterFilePath	認識パラメータファイル名
RecogModelListPath	モデルリスト名
DebugText	デバッグ用テキスト情報出力フラグ（captrecog.sh のみ）
DebugDisplay	デバッグ用画像情報表示フラグ（captrecog.sh のみ）

RecognitionResultViewerComp のスクリプト設定項目	
RecogModelListPath	モデルリスト名

MultiCameraComp のスクリプト設定項目（captrecog.sh のみ）	
camera_calib_file	キャリブレーションデータファイル名
camera_setting_file	カメラ設定ファイル名

MultiDispComp のスクリプト設定項目（captrecog.sh のみ）	
image_save_mode	カメラキャプチャ画像保存（※コメント文にして無効化）

5. 補記

5.1. OpenRTM-aistのインストール方法

■OpenRTM を入れる

OpenRTM-aist のインストールは、インストールスクリプトによる方法と apt-get を用いる方法があります。ここでは、シェルスクリプトによる方法をご説明します。

1. 公式サイトより pkg_install_ubuntu.sh をダウンロードして実行する。

ダウンロードページの URL	http://www.openrtm.org/openrtm/ja/node/849#toc3
インストールスクリプトのダウンロード URL	http://openrtp.jp/openrtm/svn/OpenRTM-aist/trunk/OpenRTM-aist/build/pkg_install100_ubuntu.sh
インストール方法の解説	http://www.openrtm.org/openrtm/ja/node/1001#toc0

端末エミュレータの画面から、下記のようにコマンドを実行してください。

```
$ wget [pkg_installXXX.sh のダウンロード URL]
$ sudo sh pkg_installXXX_ubuntu.sh
```

一括インストールシェルスクリプトの実行途中で、いくつか質問をされるので、"y"あるいは"Y"を入力しながら完了させてください。

2. インストールの確認を行う

OpenRTM がインストールされたことを確認する方法は、公式サイトの下記の情報をご参照ください。

http://www.openrtm.org/openrtm/ja/node/1001#toc4

指示にしたがいコマンドを実行して、次のような結果が表示されれば OpenRTM は正常にインストールされています。

```
$ dpkg -l 'openrtm*'
 Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Installed/Config-f/Unpacked/Failed-cfg/Half-inst/t-aWait/T-pend
|/ Err?=(none)/Hold/Reinst-required/X=both-problems (Status,Err: uppercase=bad)
||/ Name         Version       Description
+++-==============-==============-============================================
ii  openrtm-aist   1.0.0-2       OpenRTM-aist, RT-Middleware distributed by A
ii  openrtm-aist-d 1.0.0-2       OpenRTM-aist headers for development
ii  openrtm-aist-d 1.0.0-2       Documentation for openrtm-aist
ii  openrtm-aist-e 1.0.0-2       OpenRTM-aist examples
```

5.2. OpenRTM-aist 1.0.0 RELEASE C++ 版のインストール

■OpenRTM 用の Eclipse を入れる

eclipse と rtm-tools を動かすための JRE(Java Runtime Environment)をインストールします。さらに eclipse を起動するためのシェルスクリプトを作成してください。

1. Eclipse のインストール

・Eclipse3.4.2+RTSE+RTCBLinux 用全部入り

（eclipse342_rtmtools100release_linux_ja.tar.gz）をダウンロードします。rtm-tools をインストールしたいディレクトリにファイルを保存してください。

ダウンロードするファイルの URL：

http://www.openrtm.org/pub/OpenRTM-aist/tools/1.0.0/eclipse342_rtmtools100release_linux_ja.tar.gz

展開方法：

```
$ cd （インストール先のディレクトリ）
$ tar  xvzf eclipse342_rtmtools100release_linux_ja.tar.gz
```

展開すると eclipse というディレクトリができます。

2. JRE のインストール

・http://www.java.com/ja/download/から JRE の Linux 自己解凍ファイルをダウンロードし、展開してください。展開されてできたディレクトリを、jre という名前に変えて eclipse ディレクトリの下へ移動してください。下記の例は、JRE version 1.6.0_26 の場合です。

展開方法：

```
$ sh  ./jre-6u26-linux-i586.bin
$ mv  jre1.6.0_26  （インストール先のディレクトリ)/jre
```

3. eclipse の起動

eclipse を起動するためのシェルスクリプトをテキストエディタで作成してください。これを Eclipse 本体のあるディレクトリに置き、rtm-tools を利用する際に実行してください。シェルスクリプトの内容は下の通りです。

```
#!/bin/sh
export GDK_NATIVE_WINDOWS=1
./eclipse -clean -vmargs -Dorg.eclipse.swt.browser.XULRunnerPath=/usr/lib/xulrunner-1.9.2.18/xulrunner
```

このシェルスクリプトを用いず直接eclipseを起動すると、起動中にプログラムが止まってしまう問題が起きます（2011 年 6 月時点）。なおシェルスクリプト中の`xulrunner-1.9.2.18`のバージョン番号（下線部）は環境によって異なる可能性があります。セキュリティ上の理由でUbuntuがアップデートされた場合にxulrunnerのバージョン番号が変わることがあります。必ず`/usr/lib/xulrunner-`で始まるディレクトリ名を確認してください。

5.3. Ubuntu10.04 環境へのインストールの注意事項

■Ubuntu10.04 環境へのインストールの注意事項

OpenRTM を使うために、Ubuntu のネットワーク設定を変更する必要があります。Ubuntu は標準で IPv6 が有効になります。その設定が OpenRTM が使用する localhost という名前の解決をさまたげるので、この設定変更が必要になります。

テキストエディタを使い、/etc/hosts ファイルの編集してください。先頭に#記号を書き加えて下記の一行をコメントアウトしてください。

変更前

```
::1         localhost ip6-localhost ip6-loopback
```

↓（#を行頭に入れる）

変更後

```
#::1        localhost ip6-localhost ip6-loopback
```

5.4. 各モジュールで必要な設定

■作業対象認識モジュール

modelList.txt はモデルを指定するファイルで、このファイルの 2 列目にモデルへのパスが記述されています。必要に応じて書き換えを行ってください。

1 列目 モデル ID（ID 番号は任意。）

2 列目 モデルへのパス（相対パスも可。認識モジュールと同じディレクトリにモデルがあれば、パスをフルに入れる必要はない。）

OpenVGR　操作手順書

Ver.0.9.0

2011 年 12 月 21 日

独立行政法人 産業技術総合研究所

知能システム研究部門　タスクビジョン研究グループ

Copyright © 2011 AIST All Rights Reserved.