

エージェントシステム
RTM/ROS相互運用
RTM 第四回

2011/6/22 (水) 13:00-14:30

矢口 裕明

今日の内容

- RTMEXTender
 - OpenRTM vs ROS
 - How to install
- 三次元ビジョン
 - 事例紹介
 - Augmented Reality
 - Kinect

RTMEXTender

- とは、OpenRTMをより扱いやすくするための拡張アプリケーションである。
- 基本的にPythonで記述されている。
- RTMEXTender =
 - OpenRTMを拡張するもの
 - OpenRTMを延命するもの

RTMEXTender導入(1)

- その前に、最低限のgitの使い方
 - git clone [uri]
 - リポジトリの複製をローカルに作成
 - git pull
 - 複製元リポジトリの変更点を手元に持ってくる
 - git commit -m "comment"
 - ローカルリポジトリにコミット
 - git checkout [filename]
 - 手元で変更したファイルを元に戻す
 - (git push)
 - 複製元にローカルの変更点を送る。今回はやる必要なし
 - git format-patch
 - 自分のリポジトリと複製元の差分のパッチを作成
- CVS,SVNとは根本的に考え方方が違うことに注意！
 - リポジトリは複製されて自分の手元にある。コミットはいくらしても他人に迷惑はかかるない。
 - オリジナルのリポジトリを変更するのはgit push。これをしない限り絶対に他人に迷惑を掛けることはない。
 - ので、安心してコミットしよう。
 - オリジナルを修正したい時はpush出来る人にパッチを送ろう。

RTMEXTender導入(2)

- 必要なライブラリのインストールを行う。
 - OpenRTM (今回はインストール済み)
 - rtctree
 - git clone <https://github.com/gbiggs/rtctree.git>
 - cd rtctree
 - python setup.py build
 - sudo python setup.py install
 - rtshell
 - git clone <https://github.com/gbiggs/rtshell.git>
 - cd rtshell
 - python setup.py build
 - sudo python setup.py install
 - .bashrcに追加
 - export RTCTREE_NAMESERVERS=localhost

RTMEXTender導入(3)

- RTMEXTenderのインストール
 - <https://github.com/hyaguchijsk/rtmext>
 - を参照のこと
 - git clone <https://github.com/hyaguchijsk/rtmext.git>
 - .bashrcに追加
 - export PATH=\${PATH}:[/path/to/rtmext]
 - export
RTM_PACKAGE_PATH=[/path/to/openrtm]/examples:[/path/to/rtmext]/common:[/path/to/rtmext]/examples
 - function rtmcd () { cd `rtmpack find \$1` }
 - source [/path/to/rtmext]/completion.bash
- .bashrcを編集したら、必ずsource ~/.bashrcすること

OpenRTM vs ROS

- そもそもこの二つはロボットアプリケーションの通信による分散システムを実現するための枠組みという共通点があるため、度々比較される、この講義でも避けて通ることはできない。
- OpenRTMは
 - 動作、容量共に軽量である。
 - Windowsや組み込み環境に対応。
 - 提供されている周辺アプリケーション、コンポーネントが圧倒的に不足している。
- ROSは
 - 動作、容量共に重量級でマシンパワーを必要とする。
 - 提供されているパッケージやアプリケーションが非常に充実している。世界中の多数の機関がオープンソースでパッケージ群を提供している。

OpenRTMで困ったこと

- コンポーネントの起動, ポート接続, アクティベーションを全て自動的に行う手段がない。
 - 接続情報だけ覚えておくことはできる。しかしホスト名などマシン依存の情報が含まれていると他のマシンで再利用できない。
 - コンポーネントの構成が複雑になればなるほどやりづらくなる。
- ポートでやり取りされる情報の解釈を一意に決めづらい
 - .idlは型を決めるだけ。その値が何を意味するかはプログラマにしかわからない。
 - ライブラリを提供出来れば解決できる。

ROSは開発者に優しい

- ROSはstack,packageという概念を持っている。
 - 基本的なメッセージの解釈方法, 取り扱うためのライブラリをもとから持っている.
 - 依存関係をつくることができる. ライブラリへのリンクが簡単.
- ROSはコマンドラインツールが非常に充実している
 - roscd <- パッケージ名がわかれればパスを解決できる.
 - rosmake <- 依存するパッケージも含めて一度にコンパイルすることができる.
 - roslaunch <- 複数のプロセスを一斉に立ちあげたり落としたりすることができる.

RTMEXTenderが目指したもの

- とにかく簡潔軽量で一般的な言語を用いて以下の二つを実装する.
- パッケージ管理機能の実現
 - パッケージ名からパスを解決出来れば, ライブラリへのリンクが書ける.
 - パッケージ同士の依存関係が解決出来れば, 連鎖的にパッケージをビルドしていく.
- プログラムランチャーの実現
 - どんな複雑なコンポーネント構成でも一行で起動し Ctrl-Cで終了できるようにする.

RTMEXTenderを使ってみよう(1)

- インストールの確認とチュートリアル
 - rtm-naming で予めネームサーバを起動させておく
- rtmpack find rtmext_opencv
 - これでrtmext_opencvのパスが解決されればインストールは正常に出来ています.
- rtmmake sample_imageview
 - 依存するパッケージも含めてmake
- rtmlaunch sample_imageview capture.xml
 - カメラが付いている人はこちら
- rtmlaunch sample_imageview capture_from_file.xml
 - カメラが付いていない人はこちら

RTMEXTenderを使ってみよう(2)

- 自分で作ったコンポーネントをRTMEXTenderに対応させるには?
 - 環境変数RTM_PACKAGE_PATHに管理するディレクトリを追加
 - パッケージディレクトリにmanifest.xmlを作る
 - ランチャーファイルを作る

RTMEXTenderにおける パッケージ管理

- A
 - B <- RTM_PACKAGE_PATH
 - C
 - D
 - E
 - F
- 上のようなディレクトリ構造の場合、パッケージとして認識されるのはC,D,E
 - manifest.xmlの有無は無関係
 - Makefileがなければrtmmakeではスキップされるのみ
- ROSのパッケージの概念とは異なる
 - ROSの場合stackという概念もあり、stackの中にはstackかpackageが入る。

依存関係の記述

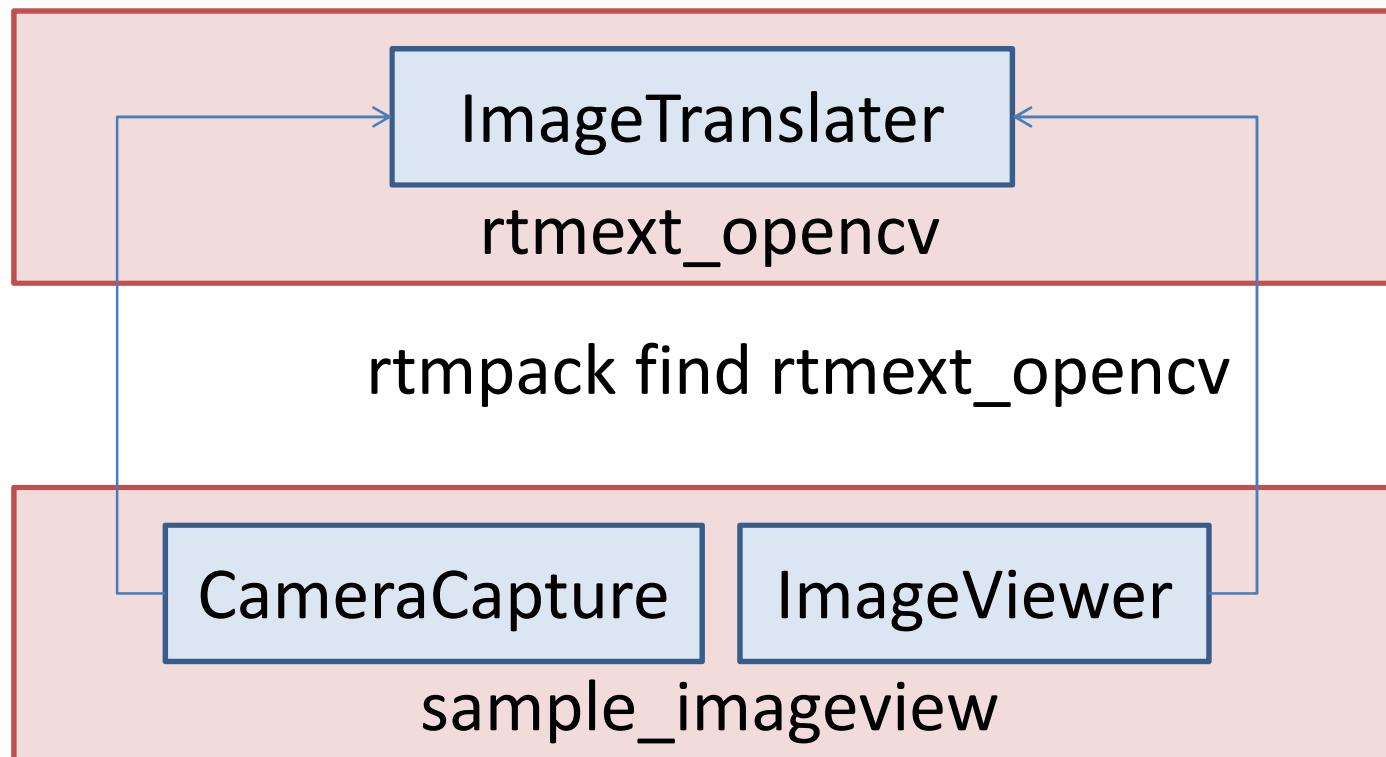
- ROSと同じフォーマットのmanifest.xmlを用いる。

```
<package>
  <description brief="brief">
    description of package
  </description>
  <author>Alan Smithee</author>
  <license>Public Domain</license>
  <depend package="dependency" />
</package>
```

ランチャーファイル

- <https://github.com/hyaguchijsk/rtmext/wiki/Format-of-rtmlaunch-.xml-file>
- xml形式での記述
- ROSと互換性は無い。
 - OpenRTMはポート間接続方式, ROSはpub/sub方式でそもそも同じように記述はできない

(参考)sample_imageviewの構造



rtmpackを駆使することでパッケージをライブラリとして使用できる

RTMEXTenderまとめ

- ソフトウェア開発はツールの提供が重要
 - どうしたらみんなが使いやすくなるかを考える。使いにくい道具はどんなにポテンシャルがあっても使われなくなってしまう。
 - 作ったら隠さず公開するが吉。ユーザーが多いということはそれだけテスターが多いということ。
 - GUIだから、コマンドラインだから優れているなどということはありえない。それぞれの良いところを積極的に利用するべき。
- これが足りないと思ったら、積極的に作って公開しましょう。
 - 暗いと不平を言うよりもすすんで明かりをつけましょう

三次元ビジョン

- カメラは三次元情報を平面に投影し、二次元画像を提供する機器である。
- カメラの二次元画像から三次元情報を復元するには？
 - ステレオカメラ：複数台のカメラを用いた三角測量
 - モーションステレオ：一台のカメラを動かして得られた複数の画像を用いた復元
 - タイムオブフライト：赤外線光が反射して戻ってくるまでの時間から距離計測
 - パターン投光：特定のパターンを照射して距離計測
 - マーカー：既知の形状、大きさの治具を設置

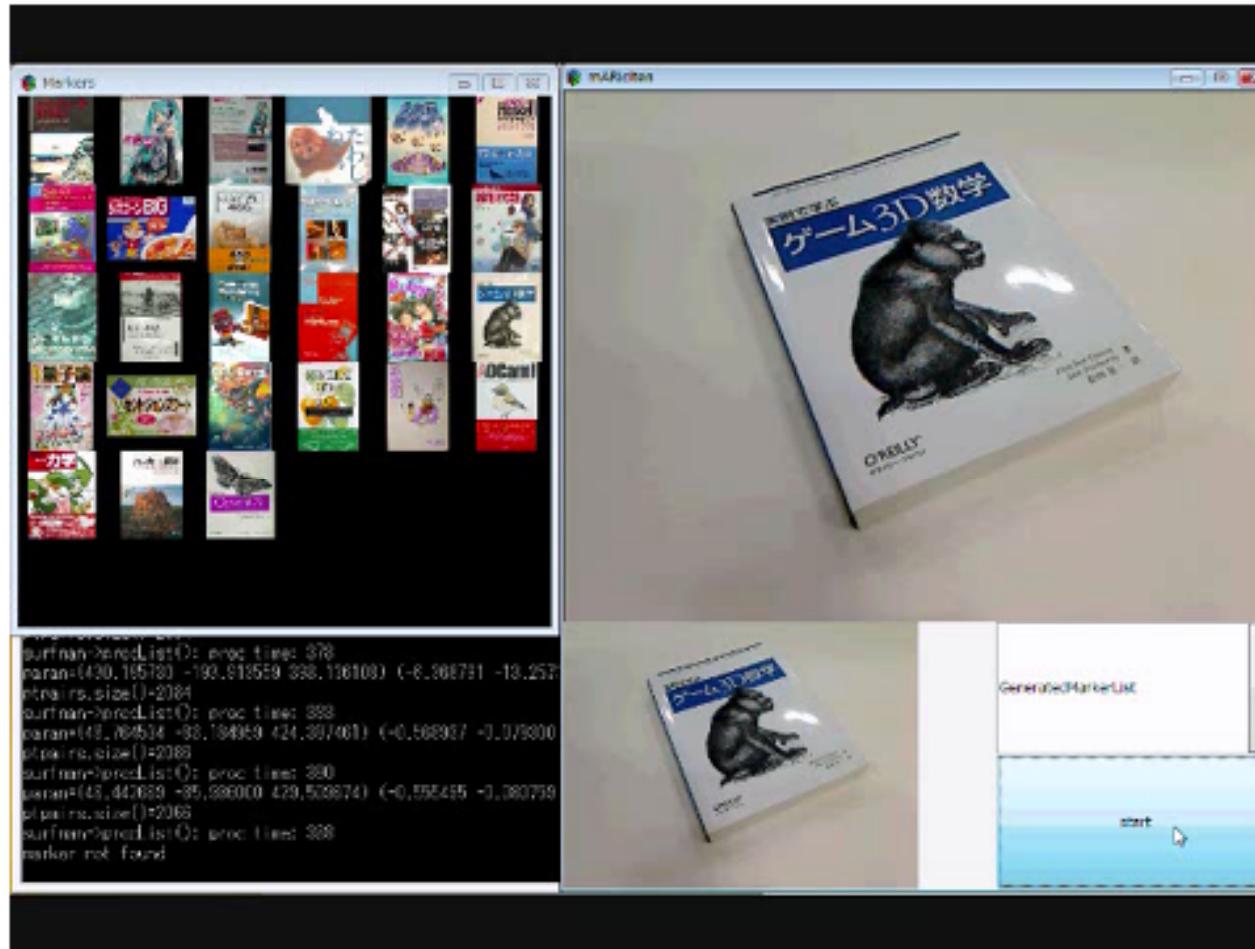
多数の画像列から三次元復元を行う

- 古くは、因子分解法と呼ばれる手法で三次元復元が行われた。
- 最近では以下の二つが有名
- 動画像からカメラの三次元姿勢と環境の復元を行うMonoSLAM
 - <http://openslam.org/robotvision.html>
- 多数の画像集合から三次元復元と画像合成を行うPhotosynth
 - <http://photosynth.net/>
 - iPhoneでも動く
<http://itunes.apple.com/jp/app/photosynth/id430065256?mt=8>
 - <http://grail.cs.washington.edu/rome/>
 - <http://grail.cs.washington.edu/projects/interior/>

Augmented Reality

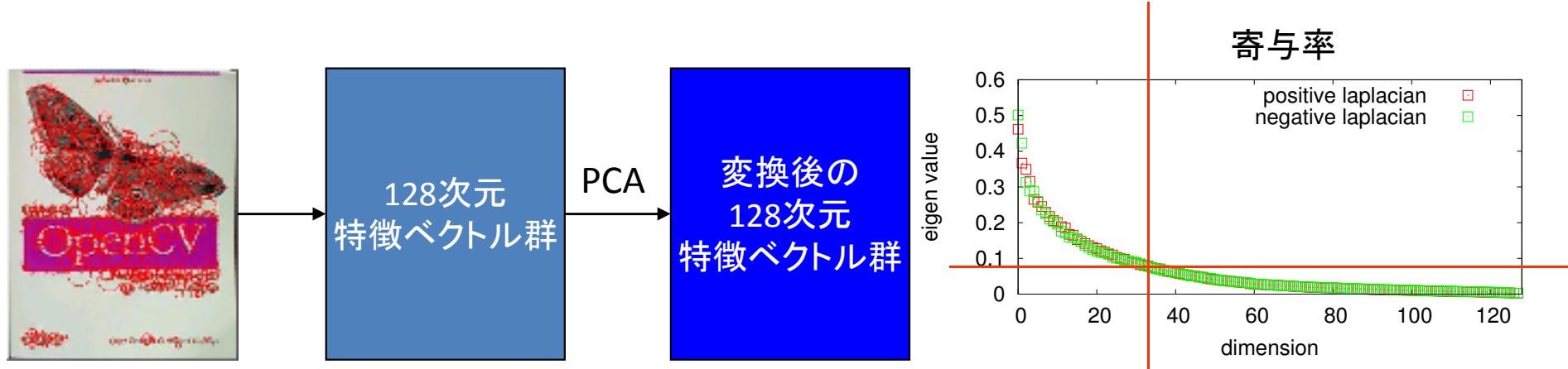
- 拡張現実. カメラ映像の中に三次元物体や付加情報を重畳して表示する。
- ターゲットとなるマーカーを環境中に設置してその三次元姿勢を推定することで三次元物体の重畳ができる
 - ARToolkit
 - <http://www.hitl.washington.edu/artoolkit/>
- MonoSLAMを応用して何も無い空間に対応
 - PTAM
 - <http://www.robots.ox.ac.uk/~gk/PTAM/>
- 商品パッケージや本の表紙などをSURFを使ってマーカーとして利用する
 - mARici-ten
 - <http://sourceforge.jp/projects/mariciten/>

高速画像マッチングと姿勢推定



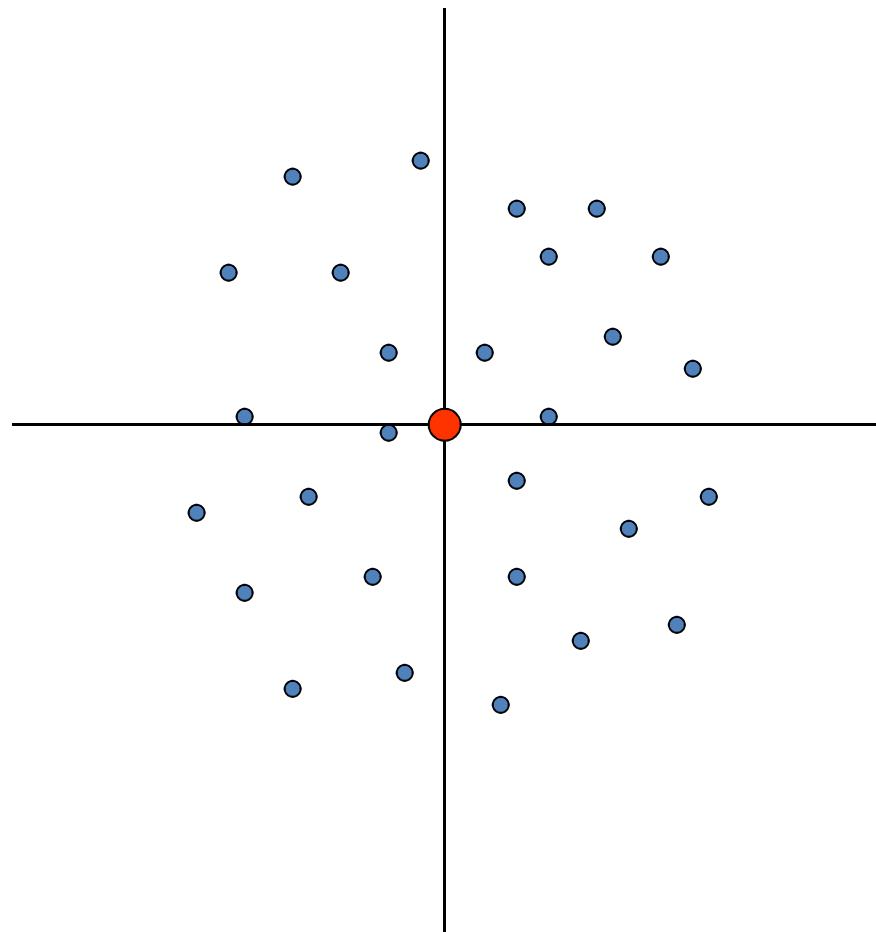
- 27枚の画像データベースを用いてカメラ映像の中にある物体のIdentificationと姿勢推定を同時に行う。
- 3fps程度で動作=0.3秒で全ての計算を終了
 - この例ではシングルコア、基本命令のみを使っている。高速化の余地あり。

高速化(1): PCAによる次元削減



- 主成分分析により、特徴ベクトルの主軸を計算しなおし、各軸の寄与率から次元削減の可能性を探る。
 - SIFTにおける先行研究では40次元程度まで削減できることがわかっている。(Ke et al, CVPR2004)
 - SURFはSIFTと同じく128次元特徴ベクトルであらわされる。ただし、ラプラシアンの符号により二系統のベクトル群が得られる。
 - ある一枚の画像から得られた二系統のベクトル群に対しPCAを施し、固有値を求めた結果、40次元付近で0.05程度となり、SIFTと同様の結果となった。
- SURF特徴ベクトルをPCAを用いて32次元まで次元削減を行う。

高速化(2): LSHによる高速探索



- 高次元における高速近似最近傍探索手法として, Locality Sensitive Hashing (Andoni et al, ACM2008)によるテーブル探索手法を用いる.
 - 全参照画像から得られた、次元削減された全特徴ベクトルの平均を取る。
 - 平均値を基準として、各次元において空間を二つに分割するようなハッシュ関数を定義する。
 - 探索時にはハッシュ関数を用いて探索する空間を絞り込む。

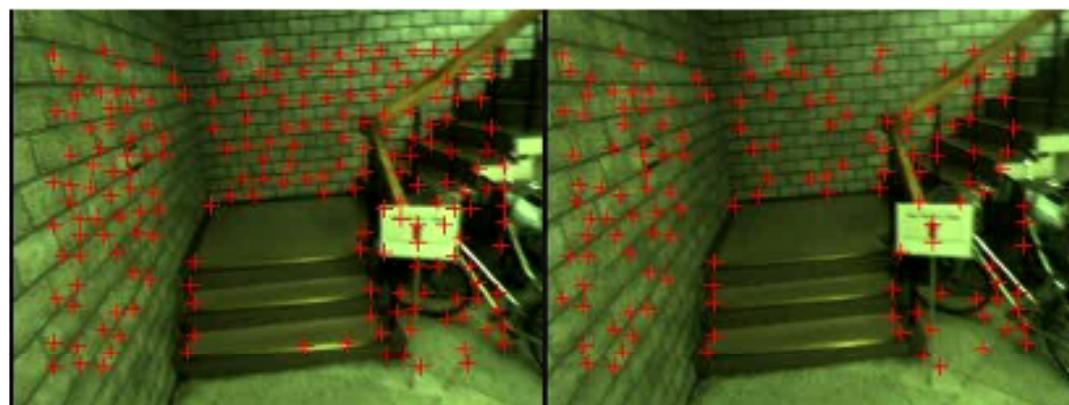
ステレオカメラを用いた環境計測

- カメラが一つで、相手が未知だと
 - スケールを決定できない=大きさの絶対値がわからない.
 - 必ず二視点以上の画像を必要とする=自分が動かないとわからない
- カメラを二つにすると
 - 大きさの絶対値がわかる.
 - 1フレームで三次元復元を行うことができる.

ビジュアルオドメトリ

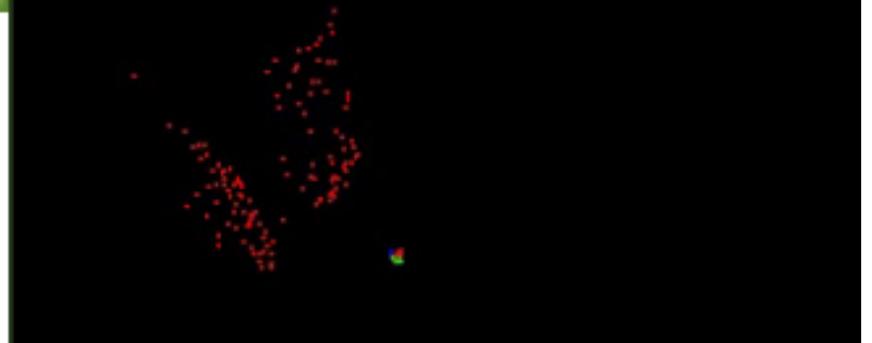
- 特徴点追跡とステレオ距離計測を用いることで三次元点の時間的な推移を知ることができる。
- 時間的に重ねあわせていくことでカメラの移動軌跡と環境の同時復元が可能。

階段の形状復元



赤い十字: 新しく選択された特徴点

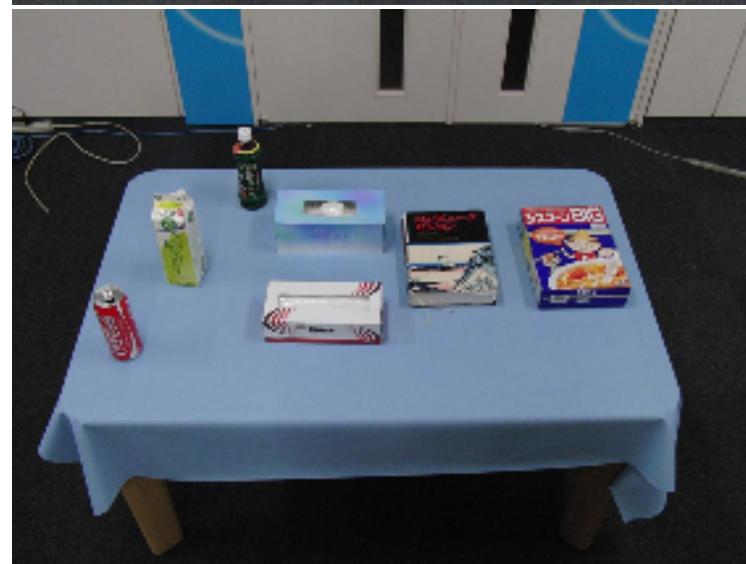
青い十字: 追跡に成功した特徴点



生活環境モデル構築

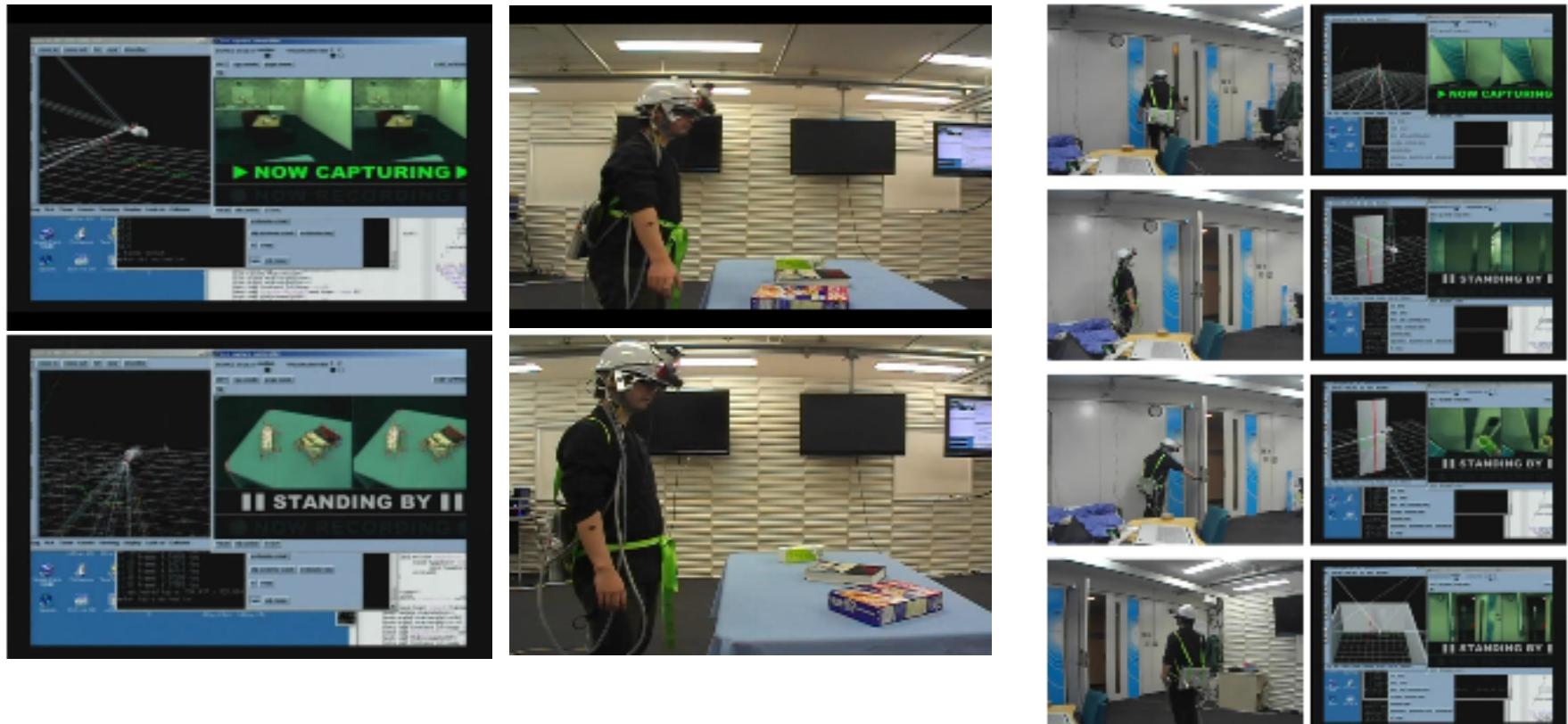


獲得したモデル



実際の物体

装着型三次元視覚による 生活環境モデリング



人間の頭部に装着したステレオカメラ等の複合センサを用いて
ボトムアップに生活環境の三次元モデルを構築していく。

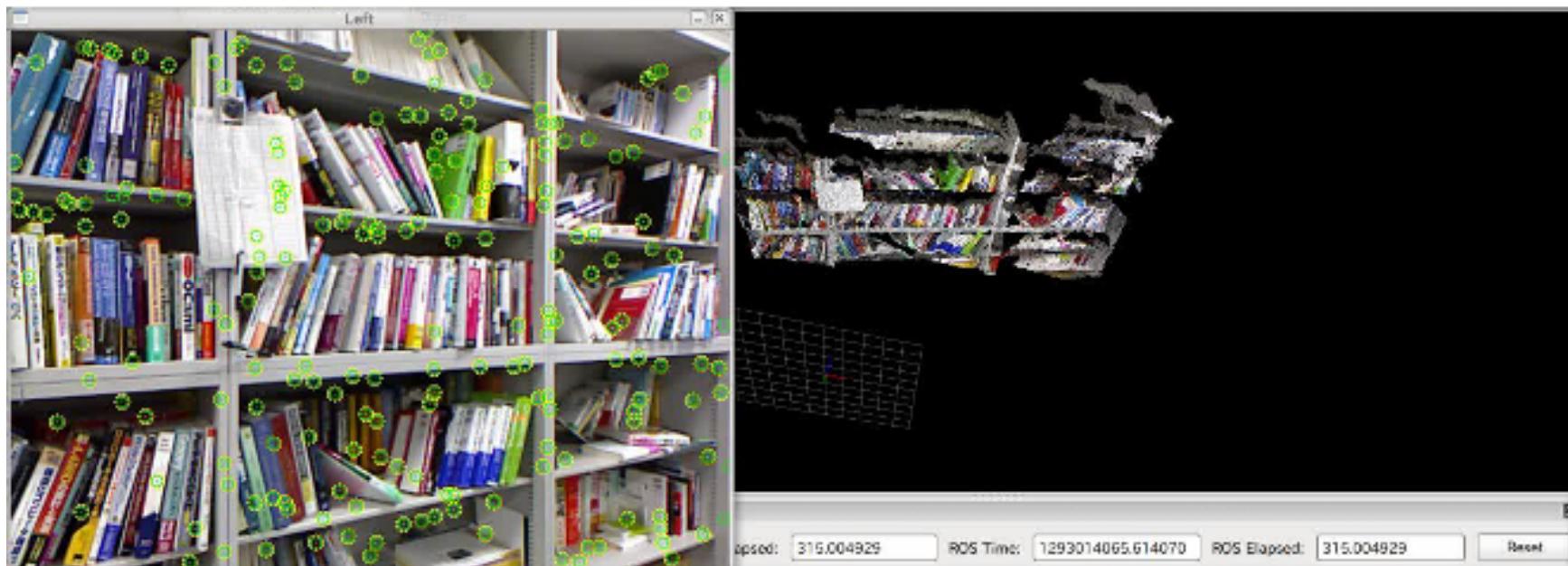
Kinect

- MicrosoftがKinectを発売(2010/11/4-20)
 - パターン投光型三次元カメラ. USB接続. 12V電源を別途必要とする. 非常に安価.
 - 原理上, ステレオカメラよりも密な三次元点群を得ることができる.
 - 煙烈を極めるハック合戦が勃発
 - 現在いくつかの方法でPCから利用可能
 - OpenKinect
 - 最初にハックされた実装.
 - OpenNI
 - 開発元のPrimeSenseが関わっている.
 - Kinect for Windows SDK beta
 - Primesenseのものに加えMicrosoftが付加した機能も利用出来る.

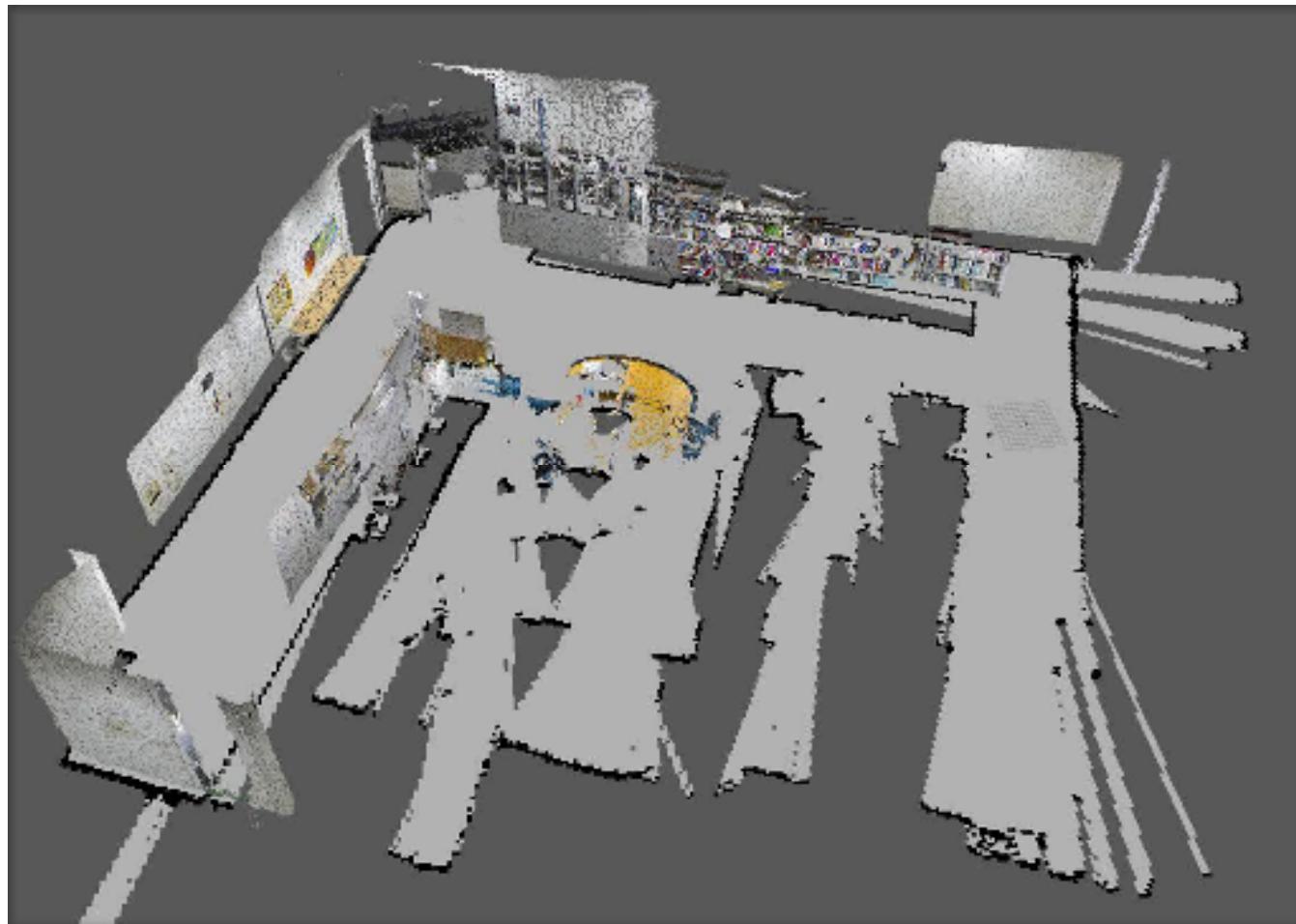
Kinectの使い方

- ROS
 - OpenNIを用いた実装がある。
 - http://www.ros.org/wiki/openni_kinect
 - 色付きの点群情報がPointCloud2でPublishされる.
- RTM
 - OpenKinectを用いた実装がある.
 - まずOpenKinectをインストール
 - <https://github.com/OpenKinect/libfreenect>
 - <https://github.com/gbiggs/rtckinect>
 - カラー画像とデプス画像が別々にポート出力される.
 - rtmext_opencvはKinectのデプス画像を距離による色分けをしたカラー画像に変換できる.

(事例紹介) Kinectを用いたビジュアルオドメトリ

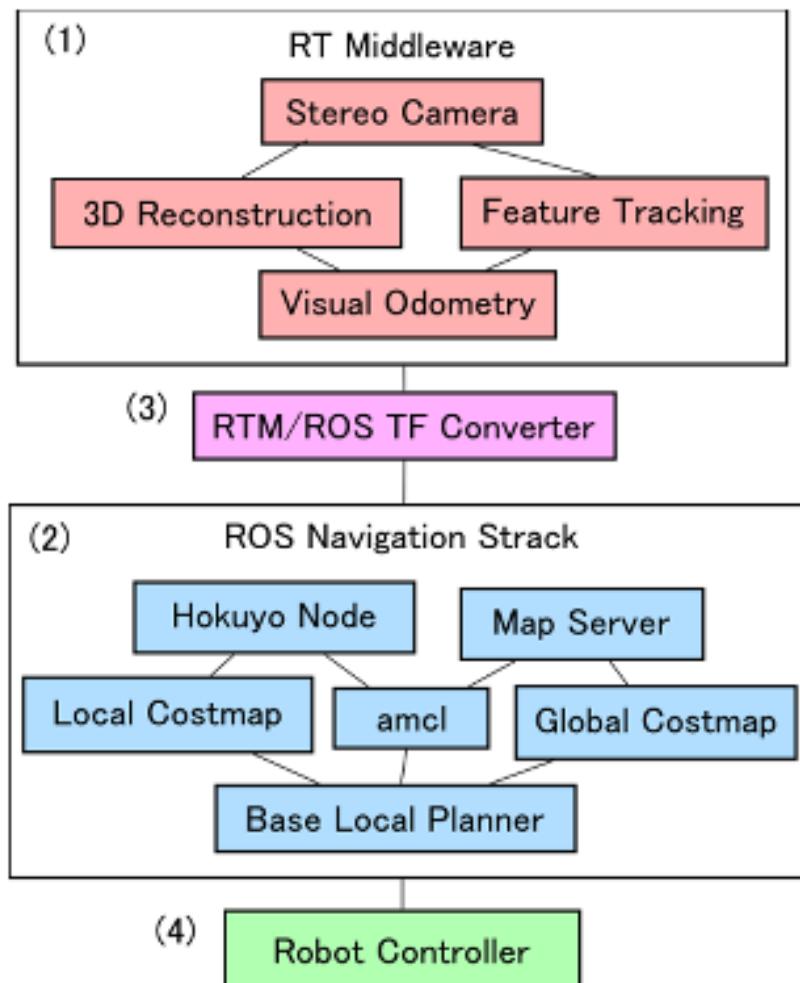


(事例紹介) Kinectを用いた三次元マップ生成



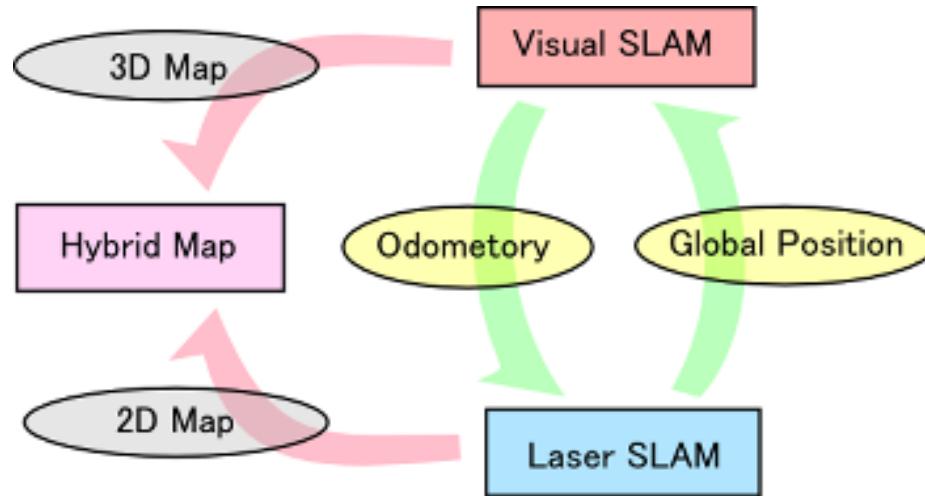
ROS 2dnav + Kinect のデプスマップ

全体システム構成



- (1) 視覚を用いた三次元復元は RTミドルウェアを用いる.
 - 視覚情報処理用のRTコンポーネントの蓄積があり、その再利用を考える.
- (2) LRFを用いた2D SLAMはROS Navigationパッケージを利用する.
 - 様々なロボットに応用されている実績がある.
- (3) 異なるフレームワークをつなぐための手法を提供する.
 - RTミドルウェアもROSも捨てることなく使いたい.

視覚と2D SLAMの接続 ROSとRTミドルウェアの接続



- 三次元情報を二次元占有格子地図に付加する.
 - ビジュアルオドメトリを初期姿勢とし, 2D SLAMでロボットの姿勢を決定する.
- 異なる二つのフレームワークを接続する.
 - RTミドルウェアのROSポートを用いてデータの送受信を行う.
 - ROSのtfを用いた座標変換をRTミドルウェアから利用する.

RGBD SLAM

- Kinectを持っている人は是非試してみてください。
 - <http://openslam.org/rgbdslam.html>

今日のまとめ

- RTMEXTender
 - OpenRTMにおける開発を支援する拡張アプリケーション
- 三次元ビジョンの事例紹介
 - できるだけ、皆さんの手元で動かせるような事例を多く紹介しました。是非試してみてください。
- 自分のソフトウェアには良い名前をつけましょう(竹内先生)

宿題

- ROSと比較してOpenRTM,RTMEXTenderになお足りない機能はなんだろうか.
 - 分析し列挙せよ.
 - [Advanced]実装してパッチを送付せよ. git format-patchで作成できる.
- RTMEXTenderを使って前回紹介したImg.idlに対応したOpenCV <-> OpenRTM相互変換ライブラリを作ってみよう.
 - 以下を参考にしよう.
 - rtmext_opencv/ImageTranslator
 - sample_imageview/CameraCapture,ImageFileCapture,ImageViewer
- 今回紹介した三次元視覚に関するソフトウェアから好きな物を一つ選んで手元のカメラで試してみよう.
 - [Advanced]ROSノードまたはOpenRTMのコンポーネントとして利用出来るようにしてみよう.