ドローン&ロボット、機械学習OSSの紹介と、OSSの品質についてのアプローチ

2017年7月12日
OSS活用WG/技術本部本部長

竹岡尚三 (株)アックス/Tier IV

今、OSS（オープンソース・ソフトウェア）なのか?

自動運転もOSS!「Autoware」

·東京大学/名古屋大学 加藤真平先生 日本で最も進んだ自動運転 研究
·名古屋大学 開発 自動運転ソフトウェア「Autoware」サポート
·名古屋大学ら、開発済み自動運転システム一式をオープンソース化…加藤准教授「時間をジャンプ」
　·Autowareは、オープンソース・ソフトウェアとして無償配布されている
http://response.jp/article/2015/08/26/258648.html

ロボット用ミドルウェア

RTミドルウェア

■産総研などが開発しているロボット用ミドルウェア
　・ RTコンポーネントは、OMGにて、国際標準化
■「OpenRTM-aist」は、RTミドルウェアの産総研による実現
　・ ライセンスは、LGPLおよび産総研と個別に契約するライセンス方式

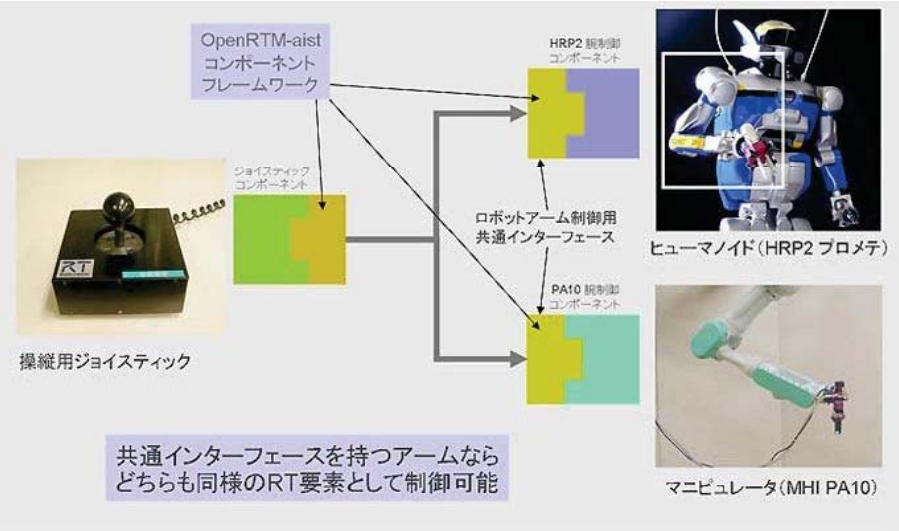
画像引用元,http://www.openrtm.org/openrtm/ja/node/161#toc5
参考
http://www.openrtm.org/openrtm/ja/content/openrtm-aist-official-website

ROS

■データの流れに応じて、コンポーネントをつなぐ
■自動車の自動運転でも採用
■OpenCVも含まれている
■雑に言ってしまえば…
　• ロボットを作るためのソフトウェア部品の多くが含まれている

画像の引用元
http://www.ros.org/news/resources/2010/poster2color_revis.jpg

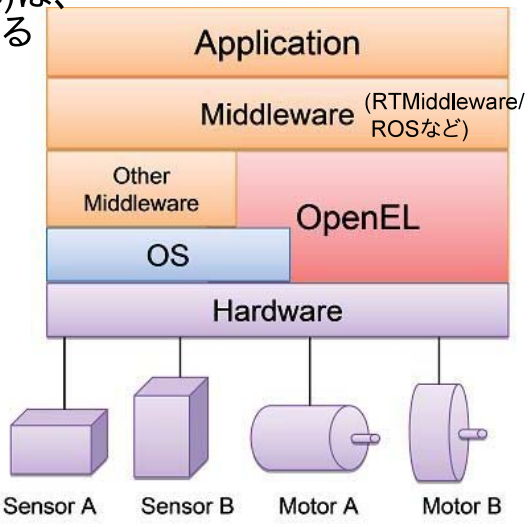
RTミドルウェア(ロボット用ミドルウェア)



引用元
http://www.aist.go.jp/aist_j/press_release/pr2005/pr20050224/pr20050224.html

OpenEL

■JASA,産総研などが推進しているフレームワーク
■国際規格にするべく、OMGに提案中
■RTミドルウェアなどのコンポーネントの可搬性を高める
■ハードウェアに近い層を、抽象化
■OpenEL準拠で書かれたソフトウェア(RTC)は、
■ハードウェア・ドライバに依存せずに動作する



参考
http://jasa.or.jp/openel/Main_Page/ja#OpenEL.E3.81.AE.E6.AD.B4.E5.8F.B2.E3.81.A8.E6.99.AE.E5.8F.8A.E3.83.BB.E5.95.93.E7.99.BA.E6.B4.BB.E5.8B.95

引用元 http://jasa.or.jp/openel/Main_Page/ja

Dronecode

https://www.dronecode.org/ Dronecode

■Linux Foundationの取りまとめで
■無人飛行体 Unmanned Aerial Vehicles (UAVs)を開発
オープンソース・ソフトウェア + オープンソース・ハードウェア
■オン・ボード(機体上)と、オフ・ボードのソフトウェアがある
- オンボード: 機体の制御、センサー制御
- オフボード: リモコン、データ収集/記録

■Dronecode は、いくつかのプロジェクトとリポジトリでできている。
• ソースコードは、下記から
Ardupilot https://github.com/diydrones/ardupilot
PX4 https://github.com/PX4
Pixhawk https://github.com/Pixhawk
MAVLink https://github.com/mavlink
UAVCAN https://github.com/uavcan
ROS https://github.com/ros
Other repositories https://github.com/Dronecode

画像引用元
https://www.dronecode.org/sites/dronecode/files/styles/dronecode_header/public/front_page_slides/images/drone_video_slide.png?itok=7C7IFYIP

IoT,M2Mもオープンソース技術

- Arduino
  - オープンソース・ソフトウェア (OSS)
    +
  - オープンソース・ハードウェア (OSHW)
  - OSS
    - 開発環境 IDE
    - ライブラリ: 膨大な量のライブラリ
    - OSSで、世界中のみんなが開発
  - OSHW
    - 回路図
    - 基板レイアウト
  - お手軽試作にバッチリ
    - ハードウェアは、アマチュア品質だが…
- Raspberry Pi
  - Linux搭載、ハードウェア情報公開

OSSの本当の利点

■「無料だから嬉しい」とか言ってると、負ける

■特定企業のOSとは違い、ソースがあるので、
理解し、独自の改良が可能
■デファクト・スタンダードがOSSなら、
特定ベンダに囲い込まれない
■自分の都合でシステムをリリースできる
- 特定のソフトウェアのリリース(バージョンアップ)に引っ張られない
- 独自に品質を上げたソフトウェアを、自由に維持できる

一般社団法人 組込みシステム技術協会 Japan Embedded Systems Technology Association

OSSに乗るしかない!

- 顧客が、OSS前提で仕様を決めている
- サービス主導の時代→OSSで素早くシステム構築
  - Windows+ブラウザで、できていることは、できて欲しい
- 高度なロボットを、すぐに作りたい
- 人工知能(AI)が入った機器をすぐに作りたい

- OSSを活用する世界をみんなで考えよう
  → JASA　OSS活用WG

一般社団法人 組込みシステム技術協会 Japan Embedded Systems Technology Association

機械学習(AI)
OSS

OSS機械学習ロボット

■人間の顔を見つけたら、追尾する
- 顔の位置を判断して、右へ行くか、左へ行くかを決める
- 阪急電車も判別可能(機械学習によって)

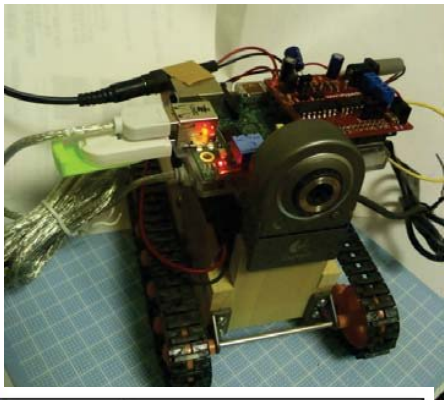
■Linux+OpenCV+OpenEL

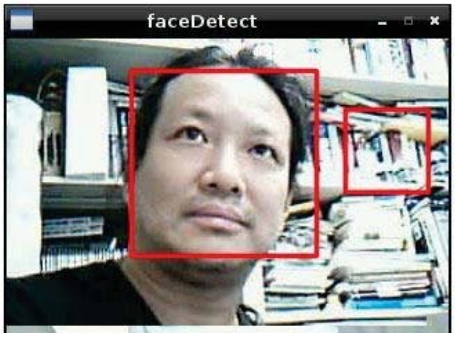
■OpenCVは
- 機械学習(SVM)による顔認識などを含む

■総合的な画像処理ライブラリ

■DCモータの制御は**OpenEL**
- OpenELはJASAなどが推進している
  ロボット用の下位ハードウェアの
  抽象化層の規格

■ハードウェア
- RaspberryPi
  ーARM11@700MHz
  ー512MBytesRAM
- USBカメラ
- DCモータ





組込みシステム技術協会
Japan Embedded Systems Technology Association

機械学習はDeep Learning だけじゃない!

■深層学習 (Deep Learning)
- 多層パーセプトロン(Perceptron)の強化形
  - パーセプトロンは、1950年代末期からある
  - バック・プロパゲーションを色んな層に
- 良い学習のためには、演算が多い
- 学習結果、判定結果は解析不能
  - 本質的に。

■サポート・ベクタ・マシン (Support Vector Machine)
- 試料をベクトル化して、比較
  - ベクトルの要素は、機械が決定する。が、人間がベクトルを見て、解析することは可能
- Deep Learningと比べて演算が少ない

■Google自動運転でDeep Learningが有名になるまでの10年間ほどは、SVMがとても多く使われていた

組み込みで使える Deep Learning(深層学習) OSS

- 組み込みで使えるものを紹介
  - 独立して動作
    - クラウドとかサーバは不要
  - 割と新しいもの
  - でも、学習時には、GPUが必要かも…

- **TensorFlow (Deep Learning OSS)**
  - Googleが開発
  - Apache 2.0ライセンス
  - AndroidやiOSでも動作
  - Pythonから使いやすい
  - https://www.tensorflow.org/

- **Chainer (Deep Learning OSS)**
  - 日本の Preferred Infrastructure 社が開発
  - ライセンス
    - 基本的に、改変、再配布など可能。詳細は下記
    - https://github.com/pfnet/chainer/blob/master/LICENSE
  - Pythonから使いやすい
  - X86 CPU以外で動作している情報が無い…
  - http://chainer.org/
- **Caffe (Deep Learning OSS)**
  - UC BerkeleyのBerkeley Vision and Learning Center (BVLC)が開発
  - 速い
  - Pythonから使える
  - http://caffe.berkeleyvision.org/

Support Vector Machine(SVM) OSS

- SVMは、計算が少ない
- ARMなどの組み込みCPUで十分に動作可能
- SVM-Light
  - Thorsten Joachims @Cornell University が開発
  - Linear kernelの学習時間は短い
  - 精度はいい
  - http://www.cs.cornell.edu/People/tj/svm_light/index.html
- SVM-perf
  - Thorsten Joachims @Cornell University が開発
  - 学習時間は短い
  - 精度はいまいち
  - http://www.cs.cornell.edu/People/tj/svm_light/svm_perf.html
- LIBSVM
  - 国立台湾大学のChih-Chung Chang と Chih-Jen Linが開発
  - 学習時間は長い
  - Linear kernelの精度は SVM-Light並
  - http://www.csie.ntu.edu.tw/~cjlin/libsvm/
- LIBLINEAR
  - 国立台湾大学 Machine Learning Group が開発
  - 学習時間がかなり短い
  - 精度はいい
  - http://www.csie.ntu.edu.tw/~cjlin/liblinear/
- CvSVM (OpenCV)
  - OpenCVに入っている
  - よく使われている
  - http://opencv.jp/opencv-2.1/cpp/support_vector_machines.html

OSS品質評価 の アプローチ

OSSって面倒くさい？

- 品質問題
  - 適当に作られた無料のソフトウェアの品質は?
  - どこの誰だか、知らない人が作ったものでしょ…
  - OSSの品質が、とても良い、ということは、しばしばある
- 品質の押さえ方は、大事
- **OSSの品質評価の問題点**
  - 開発手法で押さえられない
    - (導入したい)OSSは開発が終了しているから
  - コードレビューにそぐわない
  - 巨大OSSだから、使用したい
    - 小さいソフトウェアなら、新たに書き下ろせる
  - 巨大OSSのコードをすべて精査するのか???
    - 通常のサイトには、その能力がない

OSSの品質評価の解決方法

- テストする
- 他者の商品ソフトウェアを購入した場合
  - 製作者である他者が、品質を保証
    - テスト結果で、品質を示すことも多い
      - 必ずしもそうではないが…

- OSSもテストして、品質を確認

OSSのテスト方法の難点

- OSSもテストして、品質を確認したい…
- OSSのテストの難点
  - 仕様がいまひとつ明確でない
  - どこを重点的にテストすればいいか判らない
    - コードをレビューしていないから、
      難しそう&重要そうな部分が判らない
      ↓
  - 境界条件テストが難しい
  - 限界値テストは、可能だろう
    - 仕様が不明確な場合もあるが

OSSのひとつのテスト方法

- 境界条件テストが難しい
- 限界値テストは、可能だろう
    - 仕様が不明確な場合もあるが
      ↓
- いろんな引数を、対象OSSに食わせる
- 異常な振る舞いをしたら、危険
  - 危険な値が、入力されないように、外で処置
    or
  - OSSを修正

OSSのひとつのテスト方法:Fuzzテスト

- いろんな引数を、対象OSSに食わせる
- Fuzzテスト
  - ファズ（英：fuzz）（予測不可能な入力データ）を与えることで意図的に例外を発生させ、その例外の挙動を確認するという方法を用いる。ファズテストと呼ばれることもある。
  - https://ja.wikipedia.org/wiki/%E3%83%95%E3%82%A1%E3%82%B8%E3%83%B3%E3%82%B0 より引用

Fuzzテストを試行

- OSS活用WGで、fuzzテストを試行する
- 対象OSS
  - OpenCV :機械学習,画像処理
    - ニーズ高い
  - OpenRTM-aist (候補)
    - 規模が大きそうなので、やや躊躇している
  - OpenEL (候補)
    - JASAが中心となって策定中の規格
    - OSS版ソフトウェアを対象に行いたい

Fuzzテスト試行により

- 品質評価の基準を探る
  - Fuzzテストの結果がどの程度だと、安心だと言えるのか

    or
  - Fuzzテストの結果からは、安心だと言えないのか
- テスト・コストを探る
  - どれぐらいテストをすると
    - 問題が発現するのか
    - 十分にテストした、と言えるのか/言えないのか?

Fuzzテスト試行コストの測り方

- テスト・コストを探る
  - 「異常がすぐわかった」を 1として、
  - 他の異常が読み取れるまでのコストを、比で表す。

OpenCVにFuzzテスト試行した

- 機械学習（認識）に関する関数をテスト

- `void cvReleaseImage( IplImage** image )`
- `void cvReleaseHaarClassifierCascade( CvHaarClassifierCascade** cascade )`
- `int cvNamedWindow( const char* name, int flags=CV_WINDOW_AUTOSIZE )`
- `CvCapture* cvCreateFileCapture( const char* filename )`
- `CvCapture* cvCreateCameraCapture( int index )`
- `int cvSetCaptureProperty( CvCapture* capture, int property_id, double value )`
- `bool CascadeClassifier::load(const string& filename)`
- `CvMemStorage* cvCreateMemStorage( int block_size=0 )`
- `IplImage* cvQueryFrame( CvCapture* capture )`
- `void detectMultiScale( const Mat& image, vector<Rect>& objects, double scaleFactor=1.1, int minNeighbors=3, int flags=0, Size minSize=Size())`
- `void cvRectangle( CvArr* img, CvPoint pt1, CvPoint pt2, CvScalar color, int thickness=1, int line_type=8, int shift=0 )`
- `void cvShowImage( const char* name, const CvArr* image )`
- `void cvReleaseMemStorage( CvMemStorage** storage )`
- `void cvReleaseCapture( CvCapture** capture )`
- `void cvDestroyWindow( const char* name )`

OpenCVにFuzzテスト試行した結果

大きな問題がありそうなもののみを、表に掲げる

関数	引数	値	結果（大きな問題がありそうなもののみ抜粋）
CvCapture* cvCreateFileCapture(const char* filename)	filename	NULL	Segmentation fault
IplImage* cvQueryFrame(CvCapture* capture)	capture	NULL	静止画使用時は黒い表示、カメラ使用時はNULLを返す
void detectMultiScale(const Mat& image, vector<Rect>& objects, double scaleFactor=1.1, int minNeighbors=3, int flags=0, Size minSize=Size())	scaleFactor	inf	呼び出してから戻ってこない
void cvShowImage(const char* name, const CvArr* image)	name	存在しないウィンドウ名	(Hankyu+Face Detect:12853): GLib-GObject-CRITICAL **: g_object_unref: assertion 'G_IS_OBJECT (object)' failedを表示し、新たなウィンドウ生成
	image	NULL	真っ黒の表示。(Hankyu+Face Detect:16287): GLib-GObject-CRITICAL **: g_object_unref: assertion 'G_IS_OBJECT (object)' failed表示。

OpenCVにFuzzテスト試行した

- ファースト・インプレッション
  - OpenCVって、なかなか品質が高い(主観的感想)
  - でも、バグを見つけたぜ!
    - 異常値の対応不備。通常使用では、問題にならない
-
- コスト、指標づくりなどの分析は、これから
- ソフトウェアごとのバグの出方の傾向を分析してみたい
- OSS活用WGへのご参加を!
  - 一緒に、OSS品質評価問題に取り組みましょう!

「ドローン&ロボット、機械学習OSSの紹介と、OSSの品質についてのアプローチ」

2017/7/12 発行

発行者　一般社団法人 組込みシステム技術協会
東京都中央区日本橋大伝馬町 6-7
TEL: 03(5643)0211 FAX: 03(5643)0212
URL:http://www.jasa.or.jp/